# INTERNATIONAL STANDARD

# ISO/IEC 7942-3

First edition
1999-02-01

# Information technology — Computer graphics and image processing — Graphical Kernel Systems (GKS) —

## Part 3:
## Audit trail

*Technologies de l'information — Infographie et traitement d'image — Système graphique Kernel (GKS) —*

*Partie 3: Historique d'expertise*



Reference number
ISO/IEC 7942-3:1999(E)

## Contents







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

Printed in Switzerland

## 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 JTC1. 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 7942-3 was prepared by Joint Technical Committee ISO/IEC JTC1, *Information technology*, Subcommittee SC24, *Computer graphics and image processing*.

ISO/IEC 7942 consists of the following parts, under the general title *Information technology – Computer graphics and image processing – Graphical Kernel System (GKS):*

*Part 1: Functional description*

*Part 2: NDC metafile*

*Part 3: Audit trail*

*Part 4: Picture part archive*

## Introduction

The GKS-94 audit trail provides a file format for capturing the sequence of GKS functions invoked by an application for subsequent playback. The file format consists of a set of elements that can be used to describe the functions invoked and their parameters.

# Information technology – Computer graphics and image processing – Graphical Kernel System (GKS) – Part 3: Audit trail

## 1 Scope

This part of ISO/IEC 7942 provides a file format for capturing the sequence of GKS functions and their parameters invoked by an application, for subsequent playback.

## 2 Normative references

The following standards contains provisions which, through reference in this text, constitute provisions of this part of ISO/IEC 7942. 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 7942 are encouraged to investigate the possibility of applying the most recent editions of the standards indicated below. Members of IEC and ISO maintain registers of currently valid International Standards.

ISO/IEC 7942-1:1994, *Information technology - Computer graphics and image processing - Graphical Kernel System (GKS)* - Part 1: Functional description

ISO/IEC 7942-2:1997, *Information technology - Computer graphics and image processing - Graphical Kernel System (GKS)* - Part 2: NDC metafile

## 3 Definitions

For the purposes of this part of ISO/IEC 7942, the definitions given in ISO/IEC 7942-1 apply.

# 4 Concepts

## 4.1 The structure of a GKS-94 audit trail

The purpose of the GKS-94 audit trail is to record the sequence of functions and their associated parameters executed by an application. Both input and output parameters are recorded.

An audit trail consists of a sequence of audit trail elements, delimited by begin audit trail and end audit trail elements. There is one audit trail element for each function defined in Part 1 of ISO/IEC 7942 which can be recorded in an audit trail.

Elements consist of an element name, derived from the name of the corresponding GKS-94 function by replacing spaces in the function name with the character '_'. The element name is followed by a parameter value list, delimited by the symbols '(' and ')'. Functions with no parameters have an empty parameter list, represented as '()'. Parameter values are separated by commas ','. Each parameter value is self-delimited. Keywords in the audit trail are case sensitive.

## 4.2 Representation of data values

### 4.2.1 Introduction

Part 1 of ISO/IEC 7942 describes the parameters of the GKS-94 functions in terms of abstract data types constructed from basic types with a number of type constructors. These same abstract data types form the basis for the representations of parameter values recorded in the audit trail. In a binding of the GKS-94 functions to a programming language, these abstract data types are given concrete representations in terms of the type structure supported by the programming language. One of the consequences of this is that the situation may arise where different programming languages represent a GKS-94 basic type in different ways, for example, a name could be represented as a string or an integer. In this audit trail, concrete representations for the basic types are not defined, the representation of basic types is left implementation dependent. The audit trail records function invocations in such a way that they can be played back by the same GKS-94 implementation; issues of portability of audit trails between different implementations of GKS-94 are not addressed by this part of ISO/IEC 7942.

### 4.2.2 Representation of constants

The simplest kind of data type in GKS-94 consists of a choice between constant values, for example:

RouteDir == NDC | BACKDROP

Constant values are represented by character strings, in this case NDC and BACKDROP.

### 4.2.3 Representation of set types

Data types which consist of sets of values of another type are defined in Part 1 of ISO/IEC 7942 using the powerset constructor, ***P***, for example:

NameSet == ***P*** GenName

means that values of type NameSet consist of sets of values, each of type GenName. Values of such types are represented by the string PSET, followed by a list of values enclosed between brackets '(' and ')' and separated by commas ','. A value of type NameSet, for example, would be represented as:

PSET(<GenName>, <GenName>, ... <GenName>)

where <GenName> denotes the representation of a value of type <GenName>.

### 4.2.4 Representation of tuple types

Ordered tuples are defined in Part 1 of ISO/IEC 7942 using the Cartesian product constructor '×'. For example:

FontPrec == FontInd × Prec

defines values of the type FontPrec to consist of pairs of values, the first of type FontInd, the second of type Prec. In an audit trail, a value of such a type is represented by a string denoting the type of the value, followed by a list of the values of its components, delimited by '(' ')' and separated by a separator. In the grammar of clause 5, the symbol × is used to denote the separator, to emphasize the relationship to Part 1 of this International Standard. An implementation of the audit trial may choose any convenient concrete representation for this separator symbol. Values of the type FontPrec, for example, are denoted by:

FontPrec(<FontInd>× <Prec>)

where <FontInd> denotes the representation of a value of type FontInd and <Prec> denotes the representation of a value of type <Prec>.

### 4.2.5 Representation of function types

Functions are defined in Part 1 of this International Standard using the notation:

Selects == SelectCritType → SelectCrit

This defines Selects as a function from values of type SelectCritType to values of type SelectCrit. The arrow '→' denotes a total function, meaning that there is a value of SelectCrit for every value of type SelectCritType. The arrow '⇸' denotes a partial function, for example:

CurveBunTab == CurveInd ⇸ CurveBun

A partial function is a function which may be undefined for some values of the source type. Thus CurveBunTab defines a function from values of type CurveInd to values of type CurveBun, but for some values of type CurveInd, it is possible that there is no corresponding value of type CurveBun.

Functions are in essence sets of pairs of values, the first value of a pair being a value from the domain of the function and the second the corresponding value in the range. Functions are therefore represented in the audit trail using the notation for representing sets and a variant of the notation for representing tuples. The separator for the components of a tuple type is normally the symbol '×', but when the tuple is derived from the elements of a function, the separator symbol '→' is used instead, again to emphasize the relationship to Part 1 of this International Standard. As for tuple types, any convenient symbol may be used as a concrete representation of the symbol '→'.

A value of type CurveBunTab would be represented as:

PSET(CurveBunTab(<CurveInd>→ <CurveBun>) , CurveBunTab(<CurveInd> → <CurveBun>) ... )

An empty set is denoted by:

PSET()

Part 1 of this International Standard uses a predicate notation to define a function whose source type is a subtype of some other type, for example:

Matrix23 == $\{i, j : \mathbb{N}_1, m : \mathbf{R} \mid 1 \le i \le 2, 1 \le j \le 3 \bullet (i, j) \rightarrow m\}$

This type is a subtype of:

Matrix == $\mathbb{N}_1 \times \mathbb{N}_1 \rightarrow \mathbf{R}$

which describes a matrix of dimension $p \times q$. For Matrix23 the first index of the source is restricted to the values {1,2} and the second to the values {1,2,3}. The notation $(i,j) \rightarrow m$ means that the value $m$ is associated with the ordered pair $(i,j)$ in the source type. For cases such as the type Matrix23, where the number of

elements does not vary, (in this case a value of type Matrix 23 always consists of 6 elements), values of the type are specified by enumerating the values of the elements in row first order. Thus values of the type Matrix 23, are represented as:

Matrix23 ( <real>, <real>, <real>, <real>, <real>, <real> )

where the order of the elements is:

$M_{11}$, $M_{12}$, $M_{13}$, $M_{21}$, $M_{22}$, $M_{23}$

For those types where the domain may vary, for example, colour arrays, the representation consists of two integers giving the number of rows and number of columns in the array, followed by a sequence of values, again in row first order. The representation of sequences is described below. A $2 \times 2$ array of colour indices, for example, would be represented as:

indirectarray ( $2 \times 2 \rightarrow$ SEQ ( 11 , 12 , 21 , 22 ) )

the values of the array elements have been chosen to correspond to the order in which the elements occur in the sequence of colour indices.

### 4.2.6 Representation of discriminated union types

In order to describe some of the GKS functions in a concise manner, Part 1 of this International Standard define types which may take any of the values of a number of other types. An abbreviated example is:

PrimAttrValue == pickidentifier <<PickId>> |
nameset <<NameSet>> |
textheight <<TextHt>>

This definition states that values of the type PrimAttrValue are either values of type PickId, values of type NameSet, or values of type TextHt. The constructor names 'pickidentifier', 'nameset' and 'textheight' are chosen to describe usage of the type following.

Values of such types are represented by the constructor name followed by the corresponding value, for example:

nameset(PSET(<GenName>, <GenName>, ... <GenName>))

The constructor name is lower case.

### 4.2.7 Representation of sequence types

Sequences are represented in the audit trail using the sequence constructor, for example:

SEQ(<WCPoint>, <WCPoint>, .... <WCPoint>)

where <WCPoint> denotes the representation of a value of type WCPoint.

# 5 Audit Trail Grammar

## 5.1 Notation

The following notational conventions are used in this clause.

a) Non-terminal symbols are delimited by brackets '<' and '>'.

b) The following metasymbols define productions, grouping and repetition.

| | |
|---|---|
| ::= | 'becomes', or is 'realized' as |
| { ··· }o | denotes an optional item, with zero or more occurrences |
| <A>* | zero or more occurrences of <A> |
| <A> \| <B> | exactly one of <A> or <B> |
| { } | metabrackets, used to group items in the right hand side of a production. |

## 5.2 Audit trail

The audit trail structure is defined by a grammar. An audit trail is a sequence of audit trail elements, beginning with the element BEGIN_AUDIT_TRAIL and terminated by the element END_AUDIT_TRAIL. This International Standard does not define a precise concrete representation for audit trail elements. The grammar below defines the information that needs to be encoded in each element and suggests a concrete syntax that may be used by implementations, obtained by representing each keyword and separator symbol by the corresponding character string in an appropriate character set. Implementations may also use more compact, implementation dependent, representations.

The element BEGIN_AUDIT_TRAIL is the first element in the audit trail. It only marks the start of an audit trail. It does not correspond to a GKS-94 function. It is not an abbreviation for the AUDIT function with parameter BEGIN. Similarly the END_AUDIT_TRAIL element is the last element in an audit trail and denotes the end of the audit trail. Likewise this element does not correspond to a GKS-94 function.

```
<audit trail>           ::= <begin audit trail>
                                <audit trail element>*
                            <end audit trail>
<begin audit trail>     ::= BEGIN_AUDIT_TRAIL
                                ( <version> )
<version>               ::= <integer>

<audit trail element>   ::= <OPEN GKS>
                          | <CLOSE GKS>
                          | <SAVE GKS STATE LIST>
                          | <RESTORE GKS STATE LIST>
                          | <ESCAPE>
                          | <EMERGENCY CLOSE GKS>
                          | <ROUTE>
                          | <CREATE OUTPUT PRIMITIVE>
                          | <OPEN STENCIL>
                          | <CLOSE STENCIL>
                          | <RENAME STENCIL>
                          | <DELETE STENCIL>
                          | <CREATE STENCIL FROM BOUNDARY>
                          | <CREATE STENCIL FROM CONTOURS>
                          | <SET CONTOUR ATTRIBUTE>
```

| <SET STENCIL ATTRIBUTE>
| <GET STENCIL ATTRIBUTE>
| <INSTANCE STENCIL>
| <INSTANCE STENCIL ALONG PATH>
| <INSTANCE STENCIL SEQUENCE ALONG PATH>
| <OPEN TILING>
| <CLOSE TILING>
| <RENAME TILING>
| <DELETE TILING>
| <CREATE TILING COMPONENT>
| <SET PRIMITIVE ATTRIBUTE>
| <ADD SET OF NAMES TO NAMESET>
| <REMOVE SET OF NAMES FROM NAMESET>
| <ADD SET OF SCISSORS TO SCISSOR SET>
| <REMOVE SET OF SCISSORS FROM SCISSOR SET>
| <SET WINDOW AND VIEWPORT>
| <SET VIEWPORT INPUT PRIORITY>
| <SET SCISSOR MODE>
| <SET NORMALIZATION TRANSFORMATION NUMBER>
| <DELETE PRIMITIVES FROM NDC PICTURE>
| <ADD SET OF NAMES TO NDC PICTURE>
| <REMOVE SET OF NAMES FROM NDC PICTURE>
| <SET NDC PICTURE PRIMITIVE ATTRIBUTE>
| <ADD SET OF SCISSORS TO NDC PICTURE>
| <REMOVE SET OF SCISSORS FROM NDC PICTURE>
| <REORDER NDC PICTURE>
| <COPY NDC PICTURE TO NDC METAFILE>
| <COPY NDC METAFILE PICTURE TO NDC PICTURE>
| <OPEN PICTURE PART>
| <CLOSE PICTURE PART>
| <ARCHIVE PICTURE PART>
| <RETRIEVE PICTURE PART FROM ARCHIVE>
| <REOPEN PICTURE PART>
| <APPEND PICTURE PART>
| <RENAME PICTURE PART>
| <DELETE PICTURE PART>
| <COPY PICTURE PART FROM PICTURE PART STORE>
| <COPY NDC PICTURE TO PICTURE PART STORE>
| <SET LOGICAL INPUT DEVICE MODE>
| <REQUEST INPUT>
| <SAMPLE INPUT>
| <AWAIT INPUT>
| <FLUSH DEVICE EVENTS>
| <SET FONT INDEX MAPPING>
| <GET GLYPH NAME>
| <SET CHARACTER CODE>
| <AUDIT>
| <WRITE USER RECORD TO AUDIT>
| <PLAYBACK>
| <READ ITEM FUNCTION NAME FROM AUDIT>
| <READ ITEM FROM AUDIT>
| <PROCESS AUDIT ITEM>

| <INQUIRE OPERATING STATE ENTRY>
| <INQUIRE GKS DESCRIPTION TABLE ENTRY>
| <INQUIRE GKS STATE LIST ENTRY>
| <INQUIRE INPUT QUEUE OVERFLOW>
| <EVALUATE TRANSFORMATION MATRIX>
| <ACCUMULATE TRANSFORMATION MATRIX>
| <EVALUATE CIRCLE>
| <EVALUATE ELLIPSE>
| <EVALUATE CIRCULAR ARC 3 POINT>
| <EVALUATE CIRCULAR ARC CENTRE>
| <EVALUATE ELLIPTIC ARC>
| <EVALUATE HYPERBOLIC ARC>
| <EVALUATE PARABOLIC ARC>
| <EVALUATE WC ORDINATE>
| <OPEN WORKSTATION>
| <CLOSE WORKSTATION>
| <SAVE WORKSTATION STATE LIST>
| <RESTORE WORKSTATION STATE LIST>
| <GET WORKSTATION STATUS>
| <REMOVE BACKDROP>
| <SET REPRESENTATION>
| <SET VIEW PRIORITY>
| <SET VIEW SELECTION CRITERION>
| <SET VIEW>
| <SET WORKSTATION VISUAL EFFECTS>
| <SET WORKSTATION WINDOW AND VIEWPORT>
| <DEFINE LOGICAL INPUT DEVICE>
| <INITIALIZE LOGICAL INPUT DEVICE>
| <SET WORKSTATION SELECTION CRITERION>
| <COPY REALIZED PICTURE TO REALIZED METAFILE>
| <COPY BLANK REALIZED PICTURE TO REALIZED METAFILE>
| <COPY REALIZED METAFILE PICTURE TO BACKDROP>
| <MESSAGE>
| <INQUIRE WORKSTATION STATE LIST ENTRY>
| <INQUIRE GENERIC WORKSTATION DESCRIPTION TABLE ENTRY>
| <INQUIRE SPECIFIC WORKSTATION DESCRIPTION TABLE ENTRY>
| <INQUIRE SPECIFIC WORKSTATION DESCRIPTION TABLE ENTRY STATE>
| <RESET SPECIFIC WORKSTATION DESCRIPTION TABLE ENTRY STATE>
| <GET TEXT EXTENT>
| <EVALUATE VIEW ORIENTATION MATRIX>
| <EVALUATE VIEW MAPPING MATRIX>
| <CONVERT COLOUR>
| <CREATE SEGMENT>
| <CLOSE SEGMENT>
| <RENAME SEGMENT>
| <DELETE SEGMENT>
| <DELETE SEGMENT FROM WORKSTATION>
| <ASSOCIATE SEGMENT WITH WORKSTATION>
| <COPY SEGMENT TO WORKSTATION>
| <INSERT SEGMENT>
| <SET SEGMENT ATTRIBUTE>
| <ACTIVATE WORKSTATION>

| <DEACTIVATE WORKSTATION>
| <CLEAR WORKSTATION>
| <GET MAPPED SEGMENT NAME>
| <GET MAPPED WORKSTATION IDENTIFIER>

<end audit trail> ::= END_AUDIT_TRAIL ()

## 5.3 Audit trail elements

To facilitate comparison with Part 1, the names of parameters of each element as well as their types have been retained in the description of each audit trail element. The representation of each parameter value is given by the representation of the corresponding type. The parameter name is not used in the construction of the representation. Thus, for example, the representation of the parameter.

<error file: ErrFile>

is given by the representation of values of type

<ErrFile>

Elements are listed in the order in which they occur in clauses 12 to 14 of Part 1 of this International Standard.

<OPEN GKS> ::= OPEN_GKS ( <error file: ErrFile> )
<CLOSE GKS> ::= CLOSE_GKS ()
<SAVE GKS STATE LIST>::=SAVE_GKS_STATE_LIST ( <set of entry names: ***P*** GSLEntName>,
<entry values: GSLEntName ⇸ GSLEntValue> )
<RESTORE GKS STATE LIST>::=RESTORE_GKS_STATE_LIST (
<entry values: GSLEntName ⇸ GSLEntValue> )
<ESCAPE> ::= ESCAPE (
<specific escape function identification: EscapeFuncId>,
<escape input data record: EscapeInData>,
<escape output data record: EscapeOutData> )
<EMERGENCY CLOSE GKS>::=EMERGENCY_CLOSE_GKS ( )
<ROUTE> ::= ROUTE ( <direction: RouteDir> )
<CREATE OUTPUT PRIMITIVE>::=CREATE_OUTPUT_PRIMITIVE (
<primitive parameters: PrimParam> )
<OPEN STENCIL> ::= OPEN_STENCIL ( <stencil name: StencilName>,
<inside rule: InsideRule> )
<CLOSE STENCIL> ::= CLOSE_STENCIL ()
<RENAME STENCIL>::= RENAME_STENCIL ( <old stencil name: StencilName>,
<new stencil name: StencilName> )
<DELETE STENCIL> ::= DELETE_STENCIL ( <stencil name: StencilName> )
<CREATE STENCIL FROM BOUNDARY>::=CREATE_STENCIL_FROM_BOUNDARY (
<stencil name: StencilName>,
<inside rule: InsideRule>,
<boundary: SeqBoundary> )
<CREATE STENCIL FROM CONTOURS>::=CREATE_STENCIL_FROM_CONTOURS (
<stencil name: StencilName>,
<inside rule: InsideRule>,
<sequence of paths: SeqPaths> )
<SET CONTOUR ATTRIBUTE>::=SET_CONTOUR_ATTRIBUTE ( <contour attribute value: ContourAttrValue> )
<SET STENCIL ATTRIBUTE>::=SET_STENCIL_ATTRIBUTE (
<stencil name: StencilName>,
<stencil attribute value: StencilAttrValue> )
<GET STENCIL ATTRIBUTE>::=GET_STENCIL_ATTRIBUTE ( <stencil name: StencilName>,
<stencil attribute name: StencilAttrName>,
<error indicator: ErrInd>,

```
                        <attribute value: StencilAttrValue> )
<INSTANCE STENCIL>   ::=INSTANCE_STENCIL ( <stencil name: StencilName>,
                        <instance name: StencilName>,
                        <stencil transformation matrix: StencilTran>,
                        <instance specifier: InstanceSpec> )
<INSTANCE STENCIL ALONG PATH>::=INSTANCE_STENCIL_ALONG_PATH (
                        <stencil name: StencilName>,
                        <instance name: StencilName>,
                        <start map: LCPoint>,
                        <end map: LCPoint>,
                        <instance number: InstanceNum>,
                        <path definition: Path> )
<INSTANCE STENCIL SEQUENCE ALONG PATH>::=
                        INSTANCE_STENCIL_SEQUENCE_ALONG_PATH (
                        <instance sequence specifier: InstanceSeqSpec>,
                        <path specifier: PathSpec>,
                        <stencil transformation matrix: StencilTran> )
<OPEN TILING>    ::= OPEN_TILING ( <tiling name: TilingName> )
<CLOSE TILING>   ::= CLOSE_TILING ()
<RENAME TILING>  ::= RENAME_TILING ( <old tiling name: TilingName>,
                        <new tiling name: TilingName> )
<DELETE TILING>  ::= DELETE_TILING ( <tiling name: TilingName> )
<CREATE TILING COMPONENT>::=CREATE_TILING_COMPONENT (
                        <primitive parameters: TilingParam>,
                        <origin: LCPoint>,
                        <replication technique: RepTech> )
<SET PRIMITIVE ATTRIBUTE>::=SET_PRIMITIVE_ATTRIBUTE (
                        <primitive attribute value: PrimAttrValue> )
<ADD SET OF NAMES TO NAMESET>::=ADD_SET_OF_NAMES_TO_NAMESET (
                        <set of names: NameSet> )
<REMOVE SET OF NAMES FROM NAMESET>::=REMOVE_SET_OF_NAMES_FROM_NAMESET (
                        <set of names: NameSet> )
<ADD SET OF SCISSORS TO SCISSOR SET>::=ADD_SET_OF_SCISSORS_TO_SCISSOR_SET (
                        <set of scissors: ScissorSet> )
<REMOVE SET OF SCISSORS FROM SCISSOR SET>::=
                        REMOVE_SET_OF_SCISSORS_FROM_SCISSOR_SET (
                        <set of scissor identifiers: ScissorIdSet> )
<SET WINDOW AND VIEWPORT>::=SET_WINDOW_AND_VIEWPORT (
                        <normalization transformation number: NormTranNum1>,
                        <window limits: WCWin>,
                        <viewport limits: NDCVp> )
<SET VIEWPORT INPUT PRIORITY>::=SET_VIEWPORT_INPUT_PRIORITY (
                        <normalization transformation number: NormTranNum>,
                        <reference transformation number: NormTranNum>,
                        <relative priority: Pri> )
<SET SCISSOR MODE>   ::=SET_SCISSOR_MODE (
                        <scissor mode: ScissorMode> )
<SET NORMALIZATION TRANSFORMATION NUMBER>::=
                        SET_NORMALIZATION_TRANSFORMATION_NUMBER (
                        <normalization transformation number: NormTranNum> )
<DELETE PRIMITIVES FROM NDC PICTURE>::=DELETE_PRIMITIVES_FROM_NDC_PICTURE (
                        <selection criterion: SelectCrit> )
```

<ADD SET OF NAMES TO NDC PICTURE>::=ADD_SET_OF_NAMES_TO_NDC_PICTURE (
<set of names: NameSet>,
<selection criterion: SelectCrit> )
<REMOVE SET OF NAMES FROM NDC PICTURE>::=
REMOVE_SET_OF_NAMES_FROM_NDC_PICTURE (
<set of names: NameSet>,
<selection criterion: SelectCrit> )
<SET NDC PICTURE PRIMITIVE ATTRIBUTE>::=SET_NDC_PICTURE_PRIMITIVE_ATTRIBUTE (
<selection criterion: SelectCrit>,
<attribute value: PrimAttrValue> )
<ADD SET OF SCISSORS TO NDC PICTURE>::=ADD_SET_OF_SCISSORS_TO_NDC_PICTURE (
<set of scissors: ScissorSet>,
<selection criterion: SelectCrit> )
<REMOVE SET OF SCISSORS FROM NDC PICTURE>::=
REMOVE_SET_OF_SCISSORS_FROM_NDC_PICTURE (
<set of scissor identifiers: ScissorIdSet>,
<selection criterion: SelectCrit> )
<REORDER NDC PICTURE>::=REORDER_NDC_PICTURE (
<source selection criterion: SelectCrit>,
<reference selection criterion: SelectCrit>,
<relative position: Position> )
<COPY NDC PICTURE TO NDC METAFILE>::=
COPY_NDC_PICTURE_TO_NDC_METAFILE (
<metafile specifier: NDCMfSpec>,
<picture identifier: PicId>,
<selection criterion: SelectCrit>,
<set of names: NameSet> )
<COPY NDC METAFILE PICTURE TO NDC PICTURE>::=
COPY_NDC_METAFILE_PICTURE_TO_NDC_PICTURE (
<metafile specifier: NDCMfSpec>,
<picture identifier: PicId>,
<set of names: NameSet> )
<OPEN PICTURE PART> ::=OPEN_PICTURE_PART ( <picture part name: PicPartName> )
<CLOSE PICTURE PART>::=CLOSE_PICTURE_PART ()
<ARCHIVE PICTURE PART>::=ARCHIVE_PICTURE_PART (
<picture part name: PicPartName>,
<archive specifier: ArSpec>,
<archive name of picture part: ArName> )
<RETRIEVE PICTURE PART FROM ARCHIVE>::=RETRIEVE_PICTURE_PART_FROM_ARCHIVE (
<archive specifier: ArSpec>,
<archive name of picture part: ArName>,
<picture part name: PicPartName> )
<REOPEN PICTURE PART>::=REOPEN_PICTURE_PART ( <picture part name: PicPartName> )
<APPEND PICTURE PART>::=APPEND_PICTURE_PART (
<source picture part name: PicPartName>,
<sink picture part name: PicPartName>,
<global transformation matrix: Matrix23>,
<global transformation mode: TransMode>,
<local transformation matrix: Matrix23>,
<local transformation mode: TransMode>,
<set of names: NameSet> )
<RENAME PICTURE PART>::=RENAME_PICTURE_PART (

```
                    <old picture part name: PicPartName>,
                    <new picture part name: PicPartName> )
<DELETE PICTURE PART>::=DELETE_PICTURE_PART (
                    <picture part name: PicPartName> )
<COPY PICTURE PART FROM PICTURE PART STORE>::=
                    COPY_PICTURE_PART_FROM_PICTURE_PART_STORE (
                    <picture part name: PicPartName>,
                    <selection criterion: SelectCrit>,
                    <global transformation matrix: Matrix23>,
                    <global transformation mode: TransMode>,
                    <local transformation matrix: Matrix23>,
                    <local transformation mode: TransMode>,
                    <set of names: NameSet>,
                    <scissor select: ScissorSelect> )
<COPY NDC PICTURE TO PICTURE PART STORE>::=
                    COPY_NDC_PICTURE_TO_PICTURE_PART_STORE (
                    <selection criterion: SelectCrit>,
                    <picture part name: PicPartName>,
                    <set of names: NameSet> )
<SET LOGICAL INPUT DEVICE MODE>::=SET_LOGICAL_INPUT_DEVICE_MODE (
                    <logical input device identifier: DevId>,
                    <operating mode: OpMode> )
<REQUEST INPUT>   ::= REQUEST_INPUT ( <logical input device identifier: DevId>,
                    <status: InStatus>,
                    <logical input value: InValue> )
<SAMPLE INPUT> ::= SAMPLE_INPUT ( <logical input device identifier: DevId>,
                    <logical input value: InValue> )
<AWAIT INPUT>   ::= AWAIT_INPUT ( <timeout: Timeout>,
                    <set of logical input device identifications and logical input values: DevIdValueSet> )
<FLUSH DEVICE EVENTS>::=FLUSH_DEVICE_EVENTS (
                    <logical input device identifier: DevId> )
<SET FONT INDEX MAPPING>::=SET_FONT_INDEX_MAPPING (
                    <font index: FontInd>,
                    <font resource name: ISO9541FontName> )
<GET GLYPH NAME>::= GET_GLYPH_NAME ( <font index: FontInd>,
                    <character code: Char>,
                    <glyph name: GlyphName> )
<SET CHARACTER CODE>::=SET_CHARACTER_CODE ( <font index: FontInd>,
                    <character code: Char>,
                    <glyph name : GlyphName> )
<AUDIT>           ::= AUDIT ( <audit identifier: AuditId>,
                    <operation: AuditOps> )
<WRITE USER RECORD TO AUDIT>::=WRITE_USER_RECORD_TO_AUDIT (
                    <audit identifier: AuditId>,
                    <audit user data record: AuditUserData> )
<PLAYBACK>        ::= PLAYBACK ( <audit identifier: AuditId>,
                    <playback operation: PlaybackOps> )
<READ ITEM FUNCTION NAME FROM AUDIT>::=
                    READ_ITEM_FUNCTION_NAME_FROM_AUDIT (
                    <audit identifier: AuditId>,
                    <function name : AuditFuncName> )
<READ ITEM FROM AUDIT>::=READ_ITEM_FROM_AUDIT (
```

<audit identifier: AuditId>,
<function name: AuditFuncName>,
<function parameters: FuncParams> )
<PROCESS AUDIT ITEM>::=PROCESS_AUDIT_ITEM (
<audit identifier: AuditId>,
<process operation: ProcessOps> )
<INQUIRE OPERATING STATE ENTRY>::=INQUIRE_OPERATING_STATE_ENTRY (
<entry name: OpStEntName>,
<error indicator: ErrInd>,
<operating state entry value: OpStEntValue> )
<INQUIRE GKS DESCRIPTION TABLE ENTRY>::=
INQUIRE_GKS_DESCRIPTION_TABLE_ENTRY (
<entry name: GDTEntName>,
<error indicator: ErrInd>,
<GKS description table entry value: GDTEntValue> )
<INQUIRE GKS STATE LIST ENTRY>::=INQUIRE_GKS_STATE_LIST_ENTRY (
<entry name: GSLEntName>,
<error indicator: ErrInd>,
<GKS state list entry value: GSLEntValue> )
<INQUIRE INPUT QUEUE OVERFLOW>::=INQUIRE_INPUT_QUEUE_OVERFLOW (
<error indicator: ErrInd>,
<logical input device identifier: DevId> )
<EVALUATE TRANSFORMATION MATRIX>::=EVALUATE_TRANSFORMATION_MATRIX (
<fixed point: WCNDCPoint>,
<shift vector: WCNDCPoint>,
<rotation angle: Rad>,
<scale factors: ScaleFactors>,
<transformation matrix: Matrix23> )
<ACCUMULATE TRANSFORMATION MATRIX>::=
ACCUMULATE_TRANSFORMATION_MATRIX (
<transformation matrix: Matrix23>,
<fixed point: WCNDCPoint>,
<shift vector: WCNDCPoint>,
<rotation angle: Rad>,
<scale factors: ScaleFactors>,
<transformation matrix: Matrix23> )
<EVALUATE CIRCLE> ::=EVALUATE_CIRCLE ( <centre: WCPoint>,
<radius: WCOrdPos>,
<circle: EllipticDisc> )
<EVALUATE ELLIPSE> ::=EVALUATE_ELLIPSE ( <ellipse specifier: EllipseSpec>,
<ellipse: EllipticDisc> )
<EVALUATE CIRCULAR ARC 3 POINT>::=EVALUATE_CIRCULAR_ARC_3_POINT (
<circular arc specifier: CircularArcSpec>,
<circular arc: ConicSection> )
<EVALUATE CIRCULAR ARC CENTRE>::=EVALUATE_CIRCULAR_ARC_CENTRE (
<centre: WCPoint>,
<start vector: Vec>,
<end vector: Vec>,
<sense flag: SenseFlag>,
<radius: WCOrdPos>,
<circular arc: ConicSection> )
<EVALUATE ELLIPTIC ARC>::=EVALUATE_ELLIPTIC_ARC (

<ellipse specifier: EllipseSpec>,
<start vector: Vec>,
<end vector: Vec>,
<elliptic arc: ConicSection> )

<EVALUATE HYPERBOLIC ARC>::=EVALUATE_HYPERBOLIC_ARC (
<hyperbola specifier: HyperbolaSpec>,
<start vector: Vec>,
<end vector: Vec>,
<hyperbolic arc: ConicSection> )

<EVALUATE PARABOLIC ARC>::=EVALUATE_PARABOLIC_ARC (
<parabola specifier: ParabolaSpec>,
<parabolic arc: ConicSection> )

<EVALUATE WC ORDINATE>::=EVALUATE_WC_ORDINATE (
<WC ordinate: WCOrd>,
<ordinate selector: OrdSelector>,
<NDC value: NDCOrd> )

<OPEN WORKSTATION>::=OPEN_WORKSTATION (
<workstation identifier: WsId>,
<workstation specifier: WsSpec>,
<workstation type: WsType> )

<CLOSE WORKSTATION>::=CLOSE_WORKSTATION (
<workstation identifier: WsId> )

<SAVE WORKSTATION STATE LIST>::=SAVE_WORKSTATION_STATE_LIST (
<workstation identifier: WsId>,
<set of entry names: ***P*** WSLEntName>,
<entry values: WSLEntName ⇸ WSLEntValue> )

<RESTORE WORKSTATION STATE LIST>::=RESTORE_WORKSTATION_STATE_LIST (
<workstation identifier: WsId>,
<entry values: WSLEntName ⇸ WSLEntValue> )

<GET WORKSTATION STATUS>::=GET_WORKSTATION_STATUS (
<workstation identifier: WsId>,
<status: WsStatus> )

<REMOVE BACKDROP> ::=REMOVE_BACKDROP (
<workstation identifier: WsId> )

<SET REPRESENTATION>::=SET_REPRESENTATION ( <workstation identifier: WsId>,
<representation: Rep> )

<SET VIEW PRIORITY> ::=SET_VIEW_PRIORITY ( <workstation identifier: WsId>,
<view index: ViewInd>,
<reference view index: ViewInd>,
<relative priority: Pri> )

<SET VIEW SELECTION CRITERION>::=SET_VIEW_SELECTION_CRITERION (
<workstation identifier: WsId>,
<view index: ViewInd>,
<selection criterion: SelectCrit> )

<SET VIEW> ::= SET_VIEW ( <workstation identifier: WsId>,
<view index: ViewInd1>,
<view orientation matrix: Matrix23>,
<view mapping matrix: Matrix23>,
<view scissor: ViewScissor> )

<SET WORKSTATION VISUAL EFFECTS>::=SET_WORKSTATION_VISUAL_EFFECTS (
<workstation identifier: WsId>,
<visual effects state: VisEffSt> )

<SET WORKSTATION WINDOW AND VIEWPORT>::=
SET_WORKSTATION_WINDOW_AND_VIEWPORT (
<workstation identifier: WsId>,
<workstation window limits: LDCWin>,
<workstation viewport limits: DCVp> )
<DEFINE LOGICAL INPUT DEVICE>::=DEFINE_LOGICAL_INPUT_DEVICE (
<logical input device identifier: DevId>,
<measure sequence: SeqMeasureId>,
<trigger set: TriggerSet>,
<initial value: InitValue> )
<INITIALIZE LOGICAL INPUT DEVICE>::=INITIALIZE_LOGICAL_INPUT_DEVICE (
<logical input device identifier: DevId>,
<initial value component: InitValue> )
<SET WORKSTATION SELECTION CRITERION>::=
SET_WORKSTATION_SELECTION_CRITERION (
<workstation identifier: WsId>,
<selection type: SelectCritType>,
<selection criterion: SelectCrit> )
<COPY REALIZED PICTURE TO REALIZED METAFILE>::=
COPY_REALIZED_PICTURE_TO_REALIZED_METAFILE (
<workstation identifier: WsId>,
<metafile specifier: RealizMfSpec>,
<picture identifier: PicId> )
<COPY BLANK REALIZED PICTURE TO REALIZED METAFILE>::=
COPY_BLANK_REALIZED_PICTURE_TO_REALIZED_METAFILE (
<workstation identifier: WsId>,
<metafile specifier: RealizMfSpec>,
<picture identifier: PicId> )
<COPY REALIZED METAFILE PICTURE TO BACKDROP>::=
COPY_REALIZED_METAFILE_PICTURE_TO_BACKDROP (
<workstation identifier: WsId>,
<metafile specifier: RealizMfSpec>,
<picture identifier: PicId> )
<MESSAGE> ::= MESSAGE ( <workstation identifier: WsId>, <message: CharString> )
<INQUIRE WORKSTATION STATE LIST ENTRY>::=
INQUIRE_WORKSTATION_STATE_LIST_ENTRY (
<workstation identifier: WsId>,
<entry name: WSLEntName>,
<type of returned values: SetReal>,
<error indicator: ErrInd>,
<workstation state list entry value: WSLEntValue> )
<INQUIRE GENERIC WORKSTATION DESCRIPTION TABLE ENTRY>::=
INQUIRE_GENERIC_WORKSTATION_DESCRIPTION_TABLE_ENTRY (
<workstation generic type: WsGenType>,
<entry name: WDTEntName>,
<error indicator: ErrInd>,
<generic workstation description table entry value: WDTEntValue> )
<INQUIRE SPECIFIC WORKSTATION DESCRIPTION TABLE ENTRY>::=
INQUIRE_SPECIFIC_WORKSTATION_DESCRIPTION_TABLE_ENTRY (
<workstation identifier: WsId>,
<entry name: WDTEntName>,
<error indicator: ErrInd>,

<entry value: WDTEntValue> )
<INQUIRE SPECIFIC WORKSTATION DESCRIPTION TABLE ENTRY STATE>::=
INQUIRE_SPECIFIC_WORKSTATION_DESCRIPTION_TABLE_ENTRY_STATE (
<workstation identifier: WsId>,
<entry name: WDTEntName>,
<error indicator: ErrInd>,
<entry state: ChangeFlag> )
<RESET SPECIFIC WORKSTATION DESCRIPTION TABLE ENTRY STATE>::=
RESET_SPECIFIC_WORKSTATION_DESCRIPTION_TABLE_ENTRY_STATE (
<workstation identifier: WsId>,
<entry name: WDTEntName> )
<GET TEXT EXTENT> ::=GET_TEXT_EXTENT ( <workstation identifier: WsId>,
<text position: WCPoint>,
<character string: CharString>,
<error indicator: ErrInd>,
<concatenation point: WCPoint>,
<text extent: TextExtent> )
<EVALUATE VIEW ORIENTATION MATRIX>::=EVALUATE_VIEW_ORIENTATION_MATRIX (
<view reference point: NDCPoint>,
<view up vector: ViewUpVec>,
<view orientation matrix: Matrix23> )
<EVALUATE VIEW MAPPING MATRIX>::=EVALUATE_VIEW_MAPPING_MATRIX (
<window limits: NDCWin>,
<viewport limits: LDCVp>,
<view mapping matrix: Matrix23> )
<CONVERT COLOUR> ::=CONVERT_COLOUR ( <workstation identifier: WsId>,
<colour specifier: ColrSpec>,
<colour model: ColrModel>,
<colour coordinates: ColrCoords>,
<conversion flag: ConversionFlag> )
<CREATE SEGMENT> ::=CREATE_SEGMENT ( <segment name: SegName> )
<CLOSE SEGMENT> ::= CLOSE_SEGMENT ()
<RENAME SEGMENT> ::=RENAME_SEGMENT ( <old segment name: SegName>,
<new segment name: SegName> )
<DELETE SEGMENT>::= DELETE_SEGMENT ( <segment name: SegName> )
<DELETE SEGMENT FROM WORKSTATION>::=DELETE_SEGMENT_FROM_WORKSTATION (
<workstation identifier: WsId>,
<segment name: SegName> )
<ASSOCIATE SEGMENT WITH WORKSTATION>::=
ASSOCIATE_SEGMENT_WITH_WORKSTATION (
<workstation identifier: WsId>,
<segment name: SegName> )
<COPY SEGMENT TO WORKSTATION>::=COPY_SEGMENT_TO_WORKSTATION (
<workstation identifier: WsId>,
<segment name: SegName> )
<INSERT SEGMENT> ::= INSERT_SEGMENT ( <segment name: SegName>,
<transformation matrix: Matrix23> )
<SET SEGMENT ATTRIBUTE>::=SET_SEGMENT_ATTRIBUTE (
<segment name: SegName>,
<segment attribute value: SegAttrValue> )
<ACTIVATE WORKSTATION>::=ACTIVATE_WORKSTATION (
<workstation identifier: WsId> )

<DEACTIVATE WORKSTATION>::=DEACTIVATE_WORKSTATION (
<workstation identifier: WsId> )
<CLEAR WORKSTATION>::=CLEAR_WORKSTATION ( <workstation identifier: WsId> )
<GET MAPPED SEGMENT NAME>::=GET_MAPPED_SEGMENT_NAME (
<segment name: SegName>,
<mapped segment name: GenName> )
<GET MAPPED WORKSTATION IDENTIFIER>::=GET_MAPPED_WORKSTATION_IDENTIFIER (
<workstation identifier: WsId>,
<mapped workstation identifier: GenName> )

## 5.4 Basic type definitions

The representation of the following basic types used in Part 1 of this International Standard is language binding and implementation dependent.

<ArName>
<ArSpec>
<AuditId>
<AuditSpec>
<AuditUserData>
<Char>
<CharString>
<ColrCharacteristics>
<ErrFile>
<EscapeFuncId>
<EscapeInData>
<EscapeOutData>
<GDPData>
<GDPId>
<GlyphName>
<InData>
<integer>
<ISO9541FontName>
<MeasureId>
<Name>
<NDCMfSpec>
<PickId>
<PicPartName>
<real>
<RealizMfSpec>
<ScissorId>
<SegName>
<StencilName>
<TilingName>
<TriggerId>
<WsGenType>
<WsId>
<WsSpec>
<WsSpecInfo>

## 5.5 Derived type definitions

All the derived types used by the parameters of the audit trail elements are listed in this section, apart from the types AuditFuncName and FuncParams which are listed in a separate section due to the size of their definitions.

<***P*** AuditId> ::= PSET ( <AuditId> {, <AuditId>}o )
<***P*** (AuditId × RecAuditFlag)> ::=PSET ( <AuditIdRecAuditFlag> {, <AuditIdRecAuditFlag>}o )
<***P*** ColrModel> ::= PSET ( <ColrModel> {, <ColrModel>}o )
<***P*** CurveType> ::= PSET ( <CurveType> {, <CurveType>}o )
<***P*** DevId> ::= PSET ( <DevId> {, <DevId>}o )
<***P*** DevIdInValue> ::= PSET ( <DevIdInValue> {, <DevIdInValue>}o )
<***P*** EdgeType> ::= PSET ( <EdgeType> {, <EdgeType>}o )
<***P*** FontPrec> ::= PSET ( <FontPrec> {, <FontPrec>}o )
<***P*** GDPId> ::= PSET ( <GDPId> {, <GDPId>}o )
<***P*** GSLEntName> ::= PSET ( <GSLEntName> {, <GSLEntName>}o )
<***P*** IntStyle> ::= PSET ( <IntStyle> {, <IntStyle>}o )
<***P*** IntStyleInd> ::= PSET ( <IntStyleInd> {, <IntStyleInd>}o )
<***P*** ISO9541FontName>::= PSET ( <ISO9541FontName> {, <ISO9541FontName>}o )
<***P*** MarkerType> ::= PSET ( <MarkerType> {, <MarkerType>}o )
<***P*** MeasureId> ::= PSET ( <MeasureId> {, <MeasureId>}o )
<***P*** PicPartName> ::= PSET ( <PicPartName> {, <PicPartName>}o )
<***P*** SegName> ::= PSET ( <SegName> {, <SegName>}o )
<***P*** SSL> ::= PSET ( <SSLEntNameSSLEntValue> {, <SSLEntNameSSLEntValue>}o )
<***P*** StencilName> ::= PSET ( <StencilName> {, <StencilName>}o )
<***P*** STSL> ::= PSET ( <STSLEntNameSTSLEntValue> {, <STSLEntNameSTSLEntValue>}o )
<***P*** TilingName> ::= PSET ( <TilingName>{, <TilingName>}o )
<***P*** TriggerId> ::= PSET ( <TriggerId> {, <TriggerId>}o )
<***P*** WsGenType> ::= PSET ( <WsGenType> {, <WsGenType>}o )
<***P*** WsId> ::= PSET ( <WsId> {, <WsId>}o )
<***P*** WSLEntName> ::= PSET ( <WSLEntName> {, <WSLEntName>}o )
<seq (DevId × InValue)> ::=SEQ ( <DevIdInValue> {, <DevIdInValue>}o )
<seqWCPoint> ::= SEQ ( <WCPoint> {, <WCPoint>}o )
<AbutSpec> ::= RIGHT | LEFT | TOP
| BOTTOM | CENTREVERT | CENTREHORIZ
| CAPLINE | BASELINE
<AreaInd> ::= <integer>
<AreaAsfs> ::= AreaAsfs ( <AsfValue> × <AsfValue> × <AsfValue> × <AsfValue> × <AsfValue>
× <AsfValue> × <AsfValue> )
<AreaBun> ::= AreaBun ( <IntStyle> × <IntStyleInd> × <ColrSpec> × <EdgeFlag> × <EdgeType>
× <EdgeWidth> × <ColrSpec> )
<AreaBunTab> ::= <AreaInd ⇸ AreaBun>
<AreaInd> ::= <integer>
<AreaInd ⇸ AreaBun> ::= PSET(<AreaIndAreaBun> {, <AreaIndAreaBun> }o )
<AreaIndAreaBun> ::= AreaIndAreaBun ( <AreaInd> → <AreaBun> )
<AsfValue> ::= BUNDLED | INDIVIDUAL
<AuditIdRecAuditFlag>::= AuditIdRecAuditFlag ( <AuditId> , <RecAuditFlag> )
<AuditOps> ::= OPEN ( <AuditSpec> ) | CLOSE | BEGIN | END
<Boundary> ::= closedpathseq ( <SeqPaths> | area ( <EllipticDisc> )
<Cap> ::= BUTTED | ROUNDED | SQUARE
<CellArray> ::= CellArray ( <WCPoint> × <WCPoint> × <ColArraySpec> )
<ChangeFlag> ::= CHANGED | NOTCHANGED
<CharAsfs> ::= CharAsfs ( <AsfValue> × <AsfValue> × <AsfValue> × <AsfValue> )
<CharBun> ::= CharBun ( <FontPrec> × <CharExpan> × <CharSpace> × <ColrSpec> )

```
<CharBunTab>          ::= <CharInd ⇸ CharBun>
<CharExpan>           ::= <real>
<CharInd>             ::= <integer>
<CharInd ⇸ CharBun>   ::= PSET(<CharIndCharBun> {, <CharIndCharBun> }o )
<CharIndCharBun>      ::= CharIndCharBun ( <CharInd> → <CharBun> )
<CharSpace>           ::= <real>
<ChoiceNum>           ::= <integer>
<ChoiceStatus>        ::= OK | NOCHOICE
<ChoiceStatusChoiceNum>::=ChoiceStatusChoiceNum ( <ChoiceStatus> × <ChoiceNum> )
<CircularArcSpec>     ::= CircularArcSpec( <WCOrd>, <WCOrd>, <WCOrd>, <WCOrd>, <WCOrd>, <WCOrd> )
<ClipInd>             ::= CLIP | NOCLIP
<ColArraySpec>        ::= indirectarry ( <integer> × <integer> → SeqColrInd )
                        |  rgbarray ( <integer> × <integer> → SeqColrRGB)
                        |  cieluvarray ( <integer> × <integer> → SeqColrCIELUV)
                        |  hsvarray ( <integer> × <integer> → SeqColrHSV)
                        |  hlsarray ( <integer> × <integer> → SeqColrHLS)
<ColrAvail>           ::= COLOUR | MONOCHROME
<ColrCoords>          ::= colrrgb ( <ColrRGB> )
                        |  colrcieluv ( <ColrCIELUV> )
                        |  colrhsv  ( <ColrHSV> )
                        |  colrhls  ( <ColrHLS> )
<ColrCIELUV>          ::= <real> , <real> , <real>
<ColrHSV>             ::= <real> , <real> , <real>
<ColrHLS>             ::= <real> , <real> , <real>
<ColrInd>             ::= <integer>
<ColrInd ⇸ ColrCoords>  ::=PSET(<ColrIndColrCoords> {, <ColrIndColrCoords> }o )
<ColrIndColrCoords>   ::= ColrIndColrCoords ( <ColrInd> → <ColrCoords> )
<ColrModel>           ::= RGB | CIELUV | HSV | HLS
<ColrRGB>             ::= <real01> , <real01> , <real01>
<ColrSpec>            ::= indirect ( <ColrInd> )
                        |  colrrgb ( <ColrRGB> )
                        |  colrcieluv ( <ColrCIELUV> )
                        |  colrhsv ( <ColrHSV> )
                        |  colrhls ( <ColrHLS> )
<ColrTab>             ::= <ColrInd ⇸ ColrCoords>
<ConicSection>        ::= ConicSection ( <Matrix33> × <WCPoint> × <WCPoint> × <SenseFlag> )
<ContourAttrName>     ::= STYLE | WIDTH |CAP | JOIN
<ContourAttrName → ContourAttrValue>::=empty |
                              PSET ( { <ContourAttrNameContrAttrValue> {, <ContourAttrNameContrAttrValue>}o }o )
<ContourAttrNameContrAttrValue>::= ContourAttrNameContrAttrValue (
                              { STYLE → style <Style> ) }
                        |  { WIDTH → width ( <NDCOrd> ) }
                        |  { CAP → cap ( <Cap> ) }
                        |  { JOIN → join ( <Join> ) }
                              } )
<ContourAttrValue>    ::= style ( <Style> )
                        |  width ( <NDCOrd> )
                        |  cap ( <Cap> )
                        |  join ( <Join> )
<ControlPoints>       ::= hcp ( <SeqHWCPoint> ) | cp ( <SeqWCPoint> )
<ConversionFlag>      ::= DONE | APPROXIMATED | NOTCONVERTED
<CurveAsfs>           ::= CurveAsfs ( <AsfValue> × <AsfValue> × <AsfValue> )
```

```
<CurveBun>          ::= CurveBun ( <CurveType> × <CurveWidth> × <ColrSpec> )
<CurveBunTab>       ::= <CurveInd ⇸ CurveBun>
<CurveInd>          ::= <integer>
<CurveInd ⇸ CurveBun>  ::=PSET(<CurveIndCurveBun> {, <CurveIndCurveBun> }o )
<CurveIndCurveBun>  ::= CurveIndCurveBun ( <CurveInd> → <CurveBun> )
<CurveWidth>        ::= <real>
<CurveType>         ::= <integer>
<DCOrd>             ::= <real>
<DCVp>              ::= DCVp ( <DCOrd> , <DCOrd> , <DCOrd> , <DCOrd> )
<DCUnits>           ::= METRES | OTHER
<Design>            ::= Design ( <StencilName> × <StencilOrigin> × <StencilTran> ×
                          <TilingName> × <TilingOrigin> × <TilingTran> )
<Det>               ::= UNDETECTABLE | DETECTABLE
<DevCompose>        ::= DevCompose ( <P MeasureId> × <P TriggerId> )
<DevId> ::=         DevId ( <WsId>× <integer>× <MeasureClass> )
<DevId ⇸ DevCompose>  ::=PSET(<DevIdDevCompose> {, <DevIdDevCompose> }o )
<DevId → InitInValue> ::= PSET(<DevIdInitInValue> {, <DevIdInitInValue> }o )
<DevId ⇸ InitInValue>   ::=PSET(<DevIdInitInValue> {, <DevIdInitInValue> }o )
<DevId ⇸ OpMode>    ::= PSET(<DevIdOpMode> {, <DevIdOpMode> }o )
<DevIdDevCompose>   ::= DevIdDevCompose ( <DevId> × <DevCompose> )
<DevIdInitInValue> ::= DevIdInitInValue( <DevId> × <InitInValue> )
<DevIdInValue>      ::= DevIdInValue ( <DevId> × <InValue> )
<DevIdOpMode>       ::= DevIdOpMode( <DevId> → <OpMode> )
<DevIdValueSet>     ::= devidvalue ( <P DevIdInValue> ) | NONE
<DispKind>          ::= VECTOR | RASTER | OTHER
<DispSize>          ::= deviceunits ( <DCOrd × DCOrd> )
                    |   rasterunits ( <integer> × <integer> )
<EchoArea>          ::= EchoArea ( <DCOrd> , <DCOrd> ,<DCOrd> ,<DCOrd> )
<EchoSwitch>        ::= ECHO | NOECHO
<EdgeFlag>          ::= OFF | ON
<EdgeType>          ::= <integer>
<EdgeWidth>         ::= <real>
<EllipseSpec>       ::= EllipseSpec ( <WCPoint> , <WCPoint> , <WCPoint> )
<EllipticDisc>      ::= EllipticDisc ( <Matrix33> )
<ErrInd>            ::= <integer>
<FillArea>          ::= SEQ ( <WCPoint> , <WCPoint> , <WCPoint> {, <WCPoint>}o )
<FontInd>           ::= <integer>
<FontInd ⇸ ISO9541FontName>::=PSET ( <FontIndISO9541FontName> {, <FontIndISO9541FontName>}o )
<FontIndISO9541FontName>::=FontIndISO9541FontName ( <FontInd> , <ISO9541FontName> )
<FontPrec>          ::= FontPrec ( <FontInd> × <Prec> )
<GDP>               ::= GDP ( <SeqWCPoint> , <GDPId> , <GDPData> )
<GDTEntName>        ::= AVAILWSGENTYPES | AVAILFONTS | DEFFONTINDMAP
<GDTEntValue>       ::= availwsgentypes ( <P WsGenType> )
                    |   availfonts ( <P ISO9541FontName> )
                    |   deffontindmap ( <FontInd ⇸ ISO9541FontName> )
<GenName>           ::= workstationidentifier ( <WsId> )
                    |   segmentname ( <SegName> )
                    |   name ( <Name> )
GKSOpSt             ::= GKCL | GKOP
<GSLEntName>        ::= ROUTEDIR
                    |   SCISSORMODE
                    |   OPENWSIDS
```

| ACTIVEWSIDS
| OPENAUDITS
| OPENPLAYBACKS
| INPUTQUEUE
| FONTINDMAP
| CURPRIMATTRS
| CURNORMTRANNUM
| NORMTRANS
| VPINPRIS
| OPENPICPARTNAME
| PICPARTNAMES
| OPENSEGNAME
| SEGNAMES
| SEGSTLISTS
| OPENSTENCILNAME
| STENCILNAMES
| STENCILSTLISTS
| CURCONTOURATTRS
| OPENTILINGNAME
| TILINGNAMES

<GSLEntName ↛ GSLEntValue>::=PSET(<GSLEntNameGSLEntValue>
{, <GSLEntNameGSLEntValue> }o )

<GSLEntNameGSLEntValue> ::= GSLEntNameGSLEntValue ( { ROUTEDIR → <RouteDir> }
| { SCISSORMODE → <ScissorMode> }
| { OPENWSIDS → <***P*** WsId> }
| { ACTIVEWSIDS → <***P*** WsId> }
| { OPENAUDITS → <***P*** (AuditId × RecAuditFlag)> }
| { OPENPLAYBACKS → <***P*** AuditId> }
| { INPUTQUEUE → <SeqDevIdInValue)> }
| { FONTINDMAP → <FontInd ↛ ISO9541FontName> }
| { CURPRIMATTRS → <PrimAttr> }
| { CURNORMTRANNUM → <NormTranNum> }
| { NORMTRANS → <NormTranNum → NormTran> }
| { VPINPRIS → <VpInPri> }
| { OPENPICPARTNAME → <OpenPicPartName> }
| { PICPARTNAMES → <***P*** PicPartName> }
| { OPENSEGNAME → <OpenSegName> }
| { SEGNAMES → <***P*** SegName> }
| { SEGSTLISTS → <***P*** SSL> }
| { OPENSTENCILNAME → <OpenStencilName> }
| { STENCILNAMES → <***P*** StencilName> }
| { STENCILSTLISTS → <***P*** STSL> }
| { CURCONTOURATTRS → <ContourAttrName → ContourAttrValue> }
| { OPENTILINGNAME → <OpenTilingName> }
| { TILINGNAMES → <***P*** TilingName> }
} )

<GSLEntValue> ::= routedir ( <RouteDir> )
| scissormode ( <ScissorMode> )
| openwsids ( <***P*** WsId> )
| activewsids ( <***P*** WsId> )
| openaudits ( <***P*** (AuditId × RecAuditFlag)> )
| openplaybacks ( <***P*** AuditId> )

```
                    |  inputqueue ( <SeqDevIdInValue)> )
                    |  fontindmap ( <FontInd ⇸ ISO9541FontName> )
                    |  curprimattrs ( <PrimAttr> )
                    |  curnormtrannum ( <NormTranNum> )
                    |  normtrans ( <NormTranNum ( NormTran> )
                    |  vpinpris ( <VpInPri> )
                    |  openpicpartname ( <OpenPicPartName> )
                    |  picpartnames ( <P PicPartName> )
                    |  opensegname ( <OpenSegName> )
                    |  segnames ( <P SegName> )
                    |  segstlists ( <P SSL> )
                    |  openstencilname ( <OpenStencilName> )
                    |  stencilnames ( <P StencilName> )
                    |  stencilstlists ( <P STSL> )
                    |  curcontourattrs ( <ContourAttrName ( ContourAttrValue> )
                    |  opentilingname ( <OpenTilingName> )
                    |  tilingnames ( <P TilingName> )
<Highl>             ::= NORMAL | HIGHLIGHTED
<HorAlign>          ::= NORMAL | LEFT | CENTRE | RIGHT
<HLCPoint>          ::= HLCPoint ( <LCOrd> , <LCOrd> , <LCOrd> )
<HWCPoint>          ::= HWCPoint ( <WCOrd> , <WCOrd> , <WCOrd> )
<HyperbolaSpec>     ::= HyperbolaSpec ( <WCPoint> , <WCPoint> , <WCPoint> )
<InitInValue>       ::= InitInValue ( <InValue > × <PET> × <EchoSwitch> × <EchoArea> × <InData> )
<InitValue>         ::= initialvalue ( <InValue> )
                    |  pet ( <PET> )
                    |  echoflag ( <EchoSwitch> )
                    |  echoarea ( <EchoArea> )
                    |  datarecord ( <InData> )
                    |  all ( <InitInValue> )
<InsideRule>        ::= EVENODD | WINDING
<InstanceNum>       ::= <real>
<InstanceSeqSpec>   ::= SEQ ( <SeqSpec> {, <SeqSpec>}0 )
<InstanceSpec>      ::= put ( <LCPoint> × <LCPoint> )
                    |  align ( <LCPoint> × <LCPoint> × <LCPoint> × <LCPoint> )
                    |  map ( <LCPoint> × <LCPoint> × <LCPoint> × <LCPoint> × <LCPoint> × <LCPoint> )
<InStatus>          ::= OK | NONE
<IntStyle>          ::= HOLLOW | SOLID | PATTERN | HATCH | EMPTY
<IntStyleInd>       ::= <integer>
<IntStyleInd ⇸ ColArraySpec>::=PSET(<IntStyleIndColArraySpec>
                          {, <IntStyleIndColArraySpec> }o )
<IntStyleIndColArraySpec>::=IntStyleIndColArraySpec( <IntStyleInd> → <ColArraySpec> )
<InValue>           ::= locatorvalue ( <NormTranNumWCPoint> )
                    |  strokevalue ( <NormTranNumseqWCPoint> )
                    |  valuatorvalue ( <real> )
                    |  choicevalue ( <ChoiceStatusChoiceNum> )
                    |  pickvalue ( <PickStatusNameSetPickId> )
                    |  stringvalue ( <CharString> )
                    |  compvalue ( <seqInValue> )
<Join>              ::= ROUND | BEVEL | mitred ( <NDCOrd> )
<Knots>             ::= SEQ ( <real> {, <real>}o )
<LCConicSection>    ::= LCConicSection ( <Matrix33> , <LCPoint> , <LCPoint> , <SenseFlag> )
<LCControlPoints>   ::= hlccp ( <SeqHLCPoint> ) | lccp ( <SeqLCPoint> )
```

<LCFillArea> ::= SEQ ( <LCPoint> , <LCPoint> , <LCPoint> {, <LCPoint>}0 )
<LCORD> ::= <real>
<LCNURB> ::= LCNurb ( <integer> , <LCControlPoints> , <SeqWeight> , <Knots> , <ParamRan> )
<LCPoint> ::= LCPoint ( <LCOrd> , <LCOrd> )
<LCPolyline> ::= PSET ( <LCPoint> , <LCPoint> {, <LCPolyline>}o >
<LDCOrd> ::= <real>
<LDCRect> ::= LDCRect ( <LDCOrd> , <LDCOrd> , <LDCOrd> , <LDCOrd> )
<LDCRectSet> ::= PSET ( <LDCRect> {, <LDCRect>} 0 )
<LCStencilOrigin> ::= <LCPoint>
<LCTilingOrigin> ::= <LCPoint>
<LDCVp> ::= LDCVp ( <LDCOrd> , <LDCOrd> , <LDCOrd> , <LDCOrd> )
<LDCWin> ::= LDCWin ( <LDCOrd> , <LDCOrd> , <LDCOrd> , <LDCOrd> )
<MarkerAsfs> ::= MarkerAsfs ( <AsfValue> × <AsfValue> × <AsfValue> )
<MarkerBun> ::= MarkerBun ( <MarkerType> × <MarkerSize> × <ColrSpec> )
<MarkerBunTab> ::= <MarkerInd ⇸ MarkerBun>
<MarkerInd> ::= <integer>
<MarkerInd ⇸ MarkerBun>::=PSET(<MarkerBun> {, <MarkerIndMarkerBun>}o )
<MarkerIndMarkerBun> ::=MarkerIndMarkerBun ( <MarkerInd> → <MarkerBun> )
<MarkerSize> ::= <real>
<MarkerType> ::= <integer>
<Matrix23> :: Matrix23 ( <real> , <real> , <real> , <real> , <real> , <real> )

(Note: order of matrix elements is $M_{11}$, $M_{12}$, $M_{13}$, $M_{21}$, $M_{22}$,$M_{23}$)

<Matrix33> :: Matrix23 ( <real> , <real> , <real> , <real> , <real> , <real> , <real> , <real> , <real> )

(Note: order of matrix elements is $M_{11}$, $M_{12}$, $M_{13}$, $M_{21}$, $M_{22}$,$M_{23}$, $M_{31}$, $M_{32}$, $M_{33}$)

<MeasureClass> ::= LOCATOR | STROKE | VALUATOR | CHOICE | PICK | STRING | COMPOSITE
<NameSet> ::= PSET ( { <GenName> {, <GenName>} o } o )
<NDCOrd> ::= <real>
<NDCPoint> ::= NDCPoint ( <NDCOrd> , <NDCOrd> )
<NDCRect> ::= NDCRect ( <NDCOrd> , <NDCOrd> , <NDCOrd> , <NDCOrd> )
<NDCRectSet> ::= PSET ( <NDCRect> {, <NDCRect>} 0 )
<NDCVp> ::= NDCVp ( <NDCOrd> , <NDCOrd> , <NDCOrd> , <NDCOrd> )
<NDCWin> ::= NDCWin ( <NDCOrd> , <NDCOrd> , <NDCOrd> , <NDCOrd> )
<NormTran> ::= NormTran ( <WCWin> , <NDCVp> )
<NormTranNum> ::= <integer>
<NormTranNum1> ::= <integer>
<NormTranNum → NormTran>::=PSET ( <NormTranNumNormTran> {, <NormTranNumNormTran>}o )
<NormTranNumNormTran>::=NormTranNumNormTran ( <NormTranNum> → <NormTran> )
<NormTranNumWCPoint> ::=NormTranNumWCPoint ( <integer> × <WCPoint> )
<NormTranNumseqWCPoint>::=NormTranNumseqWCPoint ( <integer> × <seqWCPoint> )
<NURB> ::= Nurb ( <integer> , <ControlPoints> , <SeqWeight> , <Knots> , <ParamRan> )
<OpenPicPartName> ::= picturepartname ( <PicPartName> ) | undefined
<OpenSegName> ::= segmentname ( <SegName> ) | undefined
<OpenStencilName>::= stencilname ( <StencilName> ) | undefined
<OpenTilingName> ::= tilingname ( <TilingName> ) | undefined
<OpMode> ::= REQUEST | SAMPLE | EVENT
<OpStEntName> ::= GKSOPST | PICPARTOPST | SEGOPST | STENCILOPST | TILINGOPST
<OpStEntValue> ::= gksopst ( <GKSOpSt> )
| picpartopst ( <PicPartOpSt> )
| segopst ( <SegOpSt> )
| stencilopst ( <StencilOpSt> )
| tilingopst ( <TilingOpSt> )

```
<OrdSelector>          ::= X | Y
<ParabolaSpec>         ::= ParabolaSpec ( <WCPoint> , <WCPoint> , <WCPoint> )
<ParamRan>             ::= ParamRan ( <real> × <real> )
<Path>                 ::= lcpolyline ( <SeqLCPoint> )
                       |   lcnurb ( <LCNURB> )
                       |   lcconicsection ( <LCConicSection> )
<PathSpec>             ::= PathSpec ( <AbutSpec> × <Path> × <LCPoint> )
<PatSize>              ::= PatSize ( <WCOrd> , <WCOrd> )
<PatTab>               ::= <IntStyleInd ⇸ ColArraySpec>
<PET>                  ::= <integer>
<PicPartOpSt>          ::= PPCL | PPOP
<PickStatus>           ::= OK | NOPICK
<PickStatusNameSetPickId>::=PickStatusNameSetPickId ( <PickStatus> × <NameSet> × <PickId> )
<PlaybackOps>          ::= PBOPEN (<AuditSpec> ) | PBCLOSE
<Polyline>             ::= SEQ ( <WCPoint> {, <WCPoint>}o )
<Polymarker>           ::= SEQ ( <WCPoint> {, <WCPoint>}o )
<Position>             ::= FRONT | BACK
<Prec>                 ::= STRING | CHAR | STROKE
<Pri>                  ::= HIGHER | LOWER
<PrimAttr>             ::= PSET ( <PrimAttrNameValue> {, <PrimAttrNameValue>}o )
<PrimAttrNameValue>    ::= PrimAttrNameValue ( { { PICK_IDENTIFIER → pickidentifier ( <PickId> ) }
                       |   { NAMESET →  nameset ( <NameSet> ) }
                       |   { SCISSOR_SET → scissorset ( <ScissorSet> ) }
                       |   { GLOBAL_TRANSFORMATION → globaltransformation ( <Matrix23> ) }
                       |   { LOCAL_TRANSFORMATION → localtransformation ( <Matrix23> ) }
                       |   { PATTERN_SIZE → patternsize ( <PatSize> ) }
                       |   { PATTERN_REFERENCE_POINT → patternreferencepoint ( <WCPoint> ) }
                       |   { TEXT_HEIGHT → textheight ( <TextHt> ) }
                       |   { TEXT_UP_VECTOR → textupvector ( <TextUpVec> ) }
                       |   { TEXT_SKEW_ANGLE → textskewangle ( <TextSkew> ) }
                       |   { TEXT_PATH → textpath ( <TextPath> ) }
                       |   { TEXT_ALIGNMENT → textalignment ( <TextAlign> ) }
                       |   { CURVE_INDEX → curveindex ( <CurveInd> ) }
                       |   { CURVE_ASFS → curveasfs ( <CurveAsfs> ) }
                       |   { CURVETYPE → curvetype ( <CurveType> ) }
                       |   { CURVEWIDTH_SCALE_FACTOR → curvewidthscalefactor ( <CurveWidth> ) }
                       |   { CURVE_COLOUR_SPECIFIER → curvecolourspecifier ( <ColrSpec> ) }
                       |   { MARKER_INDEX → markerindex ( <MarkerInd> ) }
                       |   { MARKER_ASFS → markerasfs ( <MarkerAsfs> ) }
                       |   { MARKERTYPE → markertype ( <MarkerType> ) }
                       |   { MARKERSIZE_SCALE_FACTOR → markersizescalefactor ( <MarkerSize> ) }
                       |   { MARKER_COLOUR_SPECIFIER → markercolourspecifier ( <ColrSpec> ) }
                       |   { AREA_INDEX → areaindex ( <AreaInd> ) }
                       |   { AREA_ASFS → areaasfs ( <AreaAsfs> ) }
                       |   { INTERIOR_STYLE → interiorstyle ( <IntStyle> ) }
                       |   { INTERIOR_STYLE_INDEX → interiorstyleindex ( <IntStyleInd> ) }
                       |   { INTERIOR_COLOUR_SPECIFIER → interiorcolourspecifier ( <ColrSpec> ) }
                       |   { EDGE_FLAG → edgeflag ( <EdgeFlag> ) }
                       |   { EDGE_TYPE → edgetype ( <EdgeType> ) }
                       |   { EDGEWIDTH_SCALE_FACTOR → edgewidthscalefactor ( <EdgeWidth> ) }
                       |   { EDGE_COLOUR_SPECIFIER → edgecolourspecifier ( <ColrSpec> ) }
                       |   { CHARACTER_INDEX → characterindex ( <CharInd> ) }
```

| { CHARACTER_ASFS → characterasfs ( <CharAsfs> ) }
| { CHARACTER_FONT_AND_PRECISION→characterfontandprecision ( <FontPrec> ) }
| { CHARACTER_EXPANSION_FACTOR → characterexpansionfactor ( <CharExpan> ) }
| { CHARACTER_SPACING → characterspacing ( <CharSpace> ) }
| { CHARACTER_COLOUR_SPECIFIER → charactercolourspecifier ( <ColrSpec> ) }

<PrimAttrValue> ::= PrimAttrValue ( { pickidentifier ( <PickId> )
| nameset ( <NameSet> )
| scissorset ( <ScissorSet> )
| globaltransformation ( <Matrix23> )
| localtransformation ( <Matrix23> )
| patternsize ( <PatSize> )
| patternreferencepoint ( <WCPoint> )
| textheight ( <TextHt> )
| textupvector ( <TextUpVec> )
| textskewangle ( <TextSkew> )
| textpath ( <TextPath> )
| textalignment ( <TextAlign> )
| curveindex ( <CurveInd> )
| curveasfs ( <CurveAsfs> )
| curvetype ( <CurveType> )
| curvewidthscalefactor ( <CurveWidth> )
| curvecolourspecifier ( <ColrSpec> )
| markerindex ( <MarkerInd> )
| markerasfs ( <MarkerAsfs> )
| markertype ( <MarkerType> )
| markersizescalefactor ( <MarkerSize> )
| markercolourspecifier ( <ColrSpec> )
| areaindex ( <AreaInd> )
| areaasfs ( <AreaAsfs> )
| interiorstyle ( <IntStyle> )
| interiorstyleindex ( <IntStyleInd> )
| interiorcolourspecifier ( <ColrSpec> )
| edgeflag ( <EdgeFlag> )
| edgetype ( <EdgeType> )
| edgewidthscalefactor ( <EdgeWidth> )
| edgecolourspecifier ( <ColrSpec> )
| characterindex ( <CharInd> )
| characterasfs ( <CharAsfs> )
| characterfontandprecision ( <FontPrec> )
| characterexpansionfactor ( <CharExpan> )
| characterspacing ( <CharSpace> )
| charactercolourspecifier ( <ColrSpec> )

<PrimParam> ::= setofpolyline ( <SetOfPolyline> )
| setofnurb ( <SetOfNURB> )
| setofconicsection ( <SetOfConicSection> )
| polymarker ( <Polymarker> )
| setoffillarea ( <SetOfFillArea> )
| setofellipticsector ( <SetOfEllipticSector> )
| setofellipticsegment ( <SetOfEllipticSegment> )
| setofellipticdisc ( <SetOfEllipticDisc> )
| setofclosednurb ( <SetOfClosedNURB> )
| text ( <Text> )

```
                    |  cellarray ( <CellArray> )
                    |  design ( <Design> )
                    |  gdp ( <GDP> )
<ProcessOps>        ::= SKIP | DO
<Rad>               ::= <real>
<RecAuditFlag>      ::= OFF | ON
<Rep>               ::= curve ( <CurveInd> × <CurveBun> )
                    |  marker ( <MarkerInd> × <MarkerBun> )
                    |  area ( <AreaInd> × <AreaBun> )
                    |  pattern ( <IntStyleInd> × <ColrArraySpec> )
                    |  character ( <CharInd> × <CharBun> )
                    |  colour ( <ColrInd> × <ColrCoords> )
                    |  undefined
<RepTech>           ::= dx ( <LCOrd> ) | dy ( <LCOrd> )
                    |  dxdy ( <LCOrd> × <LCOrd> ) | dydx ( <LCOrd> × <LCOrd> )
<RouteDir>          ::= NDC | BACKDROP
<ScaleFactors>      ::= ScaleFactors ( <real> × <real> )
<Scissor>           ::= Scissor ( <ClipInd> × <NDCRectSet> × <ShieldInd> × <NDCRectSet> )
<ScissorIdScissor> ::=  ScissorIdScissor ( <ScissorId> → <Scissor> )
<ScissorIdSet>      ::= PSET ( <ScissorId> {, <ScissorId>}o )
<ScissorMode>       ::= LOCUS | SHAPE
<ScissorSelect>     ::= CHANGE | LEAVE
<ScissorSet>        ::= PSET ( <ScissorIdScissor> {, <ScissorIdScissor>}o )
<SegAttrName ↛ SegAttrValue>::=PSET ( <SegAttrNameSegAttrValue> { , <SegAttrNameSegAttrValue>}o )
<SegAttrName>       ::= VISIBILITY
                    |  HIGHLIGHTING
                    |  DETECTABILITY
                    |  SEGMENT TRANSFORMATION
                    |  SEGMENT PRIORITY
<SegAttrNameSegAttrValue>::=SegAttrNameSegAttrValue ( { {VISIBILITY → <Vis> }
                    |  { HIGHLIGHTING → <Highl> }
                    |  { DETECTABILITY → <Det> }
                    |  { SEGMENT TRANSFORMATION →  <Matrix23> }
                    |  { SEGMENT PRIORITY → <SegPri> }
                        } )
<SegAttrValue>      ::= visibility ( <Vis> )
                    |  higlighting ( <Highl> )
                    |  detectability ( <Det> )
                    |  segmenttransformation ( <Matrix23> )
                    |  segmentpriority ( <SegPri> )
<SegOpSt>           ::= SGCL | SGOP
<SegPri>            ::= <real>
<SelectCrit>        ::= contains ( <NameSet> )
                    |  isin ( <NameSet> )
                    |  equals ( <NameSet> )
                    |  SELECTALL
                    |  REJECTALL
                    |  and ( <SelectCrit> × <SelectCrit> )
                    |  or ( <SelectCrit> × <SelectCrit> )
                    |  not ( <SelectCrit> )
<SelectCritType>    ::= DISPLAY | HIGHLIGHTING | DETECTABILITY
<Selects>           ::= Selects (
```

```
                    SelectCritTypeSelectCrit ( DISPLAY → <SelectCrit> ) ,
                    SelectCritTypeSelectCrit ( HIGHLIGHTING → <SelectCrit> ) ,
                    SelectCritTypeSelectCrit ( DETECTABILITY → <SelectCrit> ) )
<SenseFlag>          ::= CLOCKWISE | ANTICLOCKWISE
<SeqBoundary>        ::= SEQ ( <Boundary> {, <Boundary>}o )
<SeqColrCIELUV>      ::= SEQ ( <ColrCIELUV>  {, <ColrCIELUV> }o )
<SeqColrHSV>         ::= SEQ ( <ColrHSV>  {, <ColrHSV> }o )
<SeqColrHLS>         ::= SEQ ( <ColrHLS>  {, <ColrHLS> }o )
<SeqColrInd>         ::= SEQ ( <ColrInd>  {, <ColrInd> }o )
<SeqColrRGB>         ::= SEQ ( <ColrRGB>  {, <ColrRGB> }o )
<SeqDevIdInValue>    ::= SEQ ( <DevIdInValue>  {, <DevIdInValue> }o )
<SeqHLCPoint>        ::= SEQ ( <HLCPoint> {, <HLCPoint>}o )
<SeqHWCPoint>        ::= SEQ ( <HWCPoint> {, <HWCPoint>}o )
<SeqLCPoint>         ::= SEQ ( <LCPoint> {, <LCPoint>}o )
<SeqMeasureId>       ::= SEQ ( <MeasureId> {, <MeasureId>}o )
<seqInValue>         ::= SEQ ( <InValue> {, <InValue>}o )
<SeqPaths>           ::= SEQ ( <Path> {, <Path>}o )
<SeqSpec>            ::= SeqSpec ( <StencilName> × <StencilName> × <LCPoint> )
<SeqWCPoint>         ::= SEQ ( <WCPoint> {, <WCPoint>}o )
<SeqWeight>          ::= SEQ ( <Weight> {, <Weight>}o )
<SetOfClosedNURB>    ::= SEQ ( <NURB> {, <NURB>}o )
<SetOfConicSection>  ::= SEQ ( <ConicSection> {, <ConicSection>}o )
<SetOfEllipticDisc>  ::= SEQ ( <EllipticDisc> {, <EllipticDisc>}o )
<SetOfEllipticSector>  ::= SEQ ( <ConicSection> {, <ConicSection>}o )
<SetOfEllipticSegment>::= SEQ ( <ConicSection> {, <ConicSection>}o )
<SetOfFillArea>      ::= SEQ ( <FillArea> {, <FillArea>}o )
<SetOfLCConicSection>   ::=PSET ( <LCConicSection> {, <LCConicSection>}o >
<SetOfLCFillArea>    ::= PSET ( <LCFillArea> {, <LCFillArea>}o >
<SetOfLCNURB>        ::= PSET ( <LCNURB> {, <LCNURB>}o >
<SetOfLCPolyline>    ::= PSET ( <LCPolyline> {, <LCPolyline>}o >
<SetOfNURB>          ::= SEQ ( <NURB> {, <NURB>}o )
<SetOfPolyline>      ::= SEQ ( <Polyline> {, <Polyline>}o )
<SetReal>            ::= SET | REALIZED
<ShieldInd>          ::= SHIELD | NOSHIELD
<SSLEntName>         ::= SEGNAME | ASSOCWSIDS | CURSEGATTRS
<SSLEntNameSSLEntValue>::SSLEntNameSSLEntValue ( { { SEGNAME , <SegName> }
                    | { ASSOCWSIDS , <P WSId> }
                    { CURSEGATTRS , <SegAttrName →→ SegAttrValue> }
                    } )
<StencilAttrName>    ::= TOPY | CAPY | HALFY | BASEY | BOTTOMY | CENTREY
                    |   LEFTX | RIGHTX | CENTREX
                    |   CENTRE | ORIGIN | CENTRETOP | CENTREBOTTOM
                    |   CENTRELEFT | CENTRERIGHT
                    |   TOPLEFT | TOPRIGHT | BOTTOMLEFT | BOTTOMRIGHT
<StencilAttrName →→ StencilAttrValue>::=PSET ( <StencilAttrNameStencilAttrValue>
                        {, <StencilAttrNameStencilAttrValue>}o )
<StencilAttrNameStencilAttrValue>::=StencilAttrNameStencilAttrValue ( {
                    { TOPY → topy ( <LCORD> ) }
                |   { CAPY → capy ( <LCORD> ) }
                |   { HALFY → halfy( <LCORD> ) }
                |   { BASEY → basey ( <LCORD> ) }
                |   { BOTTOMY → bottomy ( <LCORD> ) }
```

```
                    |   { CENTREY → centrey ( <LCORD> ) }
                    |   { LEFTX → leftx ( <LCORD> ) }
                    |   { RIGHTX → rightx ( <LCORD> ) }
                    |   { CENTREX → centrex ( <LCORD> ) }
                    |   { CENTRE → centre ( <LCPOINT> ) }
                    |   { ORIGIN → origin ( <LCPOINT> ) }
                    |   { CENTRETOP → centretop ( <LCPOINT> ) }
                    |   { CENTREBOTTOM → centrebottom ( <LCPOINT> ) }
                    |   { CENTRELEFT → centreleft ( <LCPOINT> ) }
                    |   { CENTRERIGHT → centreright ( <LCPOINT> ) }
                    |   { TOPLEFT → topleft ( <LCPOINT> ) }
                    |   { TOPRIGHT → topright ( <LCPOINT> ) }
                    |   { BOTTOMLEFT → bottomleft ( <LCPOINT> ) }
                    |   { BOTTOMRIGHT → bottomright ( <LCPOINT> ) }
<StencilAttrValue>  ::= topy ( <LCORD> )
                    |   capy ( <LCORD> )
                    |   halfy ( <LCORD> )
                    |   basey ( <LCORD> )
                    |   bottomy ( <LCORD> )
                    |   centrey ( <LCORD> )
                    |   leftx ( <LCORD> )
                    |   rightx ( <LCORD> )
                    |   centrex ( <LCORD> )
                    |   centre ( <LCPOINT> )
                    |   origin ( <LCPOINT> )
                    |   centretop ( <LCPOINT> )
                    |   centrebottom ( <LCPOINT> )
                    |   centreleft ( <LCPOINT> )
                    |   centreright ( <LCPOINT> )
                    |   topleft ( <LCPOINT> )
                    |   topright ( <LCPOINT> )
                    |   bottomleft ( <LCPOINT> )
                    |   bottomright ( <LCPOINT> )
<StencilOpSt>       ::= STCL | STOP
<StencilOrigin>     ::= StencilOrigin ( <WCPoint> )
<StencilTran>       ::= StencilTran ( <Matrix23> )
<Style>             ::= SOLID | DASHED | DOTTED | DASHED-DOTTED
<STSLEntName>       ::= STENCILNAME | CURSTENCILATTRS
<STSLEntNameSTSTEntValue>::=STSLEntNameSTSLEntValue ( { { STENCILNAME , <StencilName> }
                    |   { CURSTENCILATTRS ,<StencilAttrName ↛ StencilAttrValue> }
                        } )
<Text>              ::= Text ( <WCPoint> × <CharString> )
<TextAlign>         ::= TextAlign ( <HorAlign> × <VertAlign> )
<TextExtent>        ::= TextExtent ( <WCPoint> × <WCPoint> × <WCPoint> × <WCPoint> )
<TextHt>            ::= TextHt ( <WCOrd> )
<TextUpVec>         ::= TextUpVec ( <WCOrd> , <WCOrd> )
<TextPath>          ::= RIGHT | LEFT | UP | DOWN
<TextSkew>          ::= TextSkew ( <real> )
<TilingOpSt>        ::= TLCL | TLOP
<TilingOrigin>      ::= TilingOrigin ( <WCPoint> )
<TilingParam>       ::= tpsetofpolyline ( <TPSetOfPolyline> )
                    |   tpsetofnurb ( <TPSetOfNURB> )
```

```
                    |  tpsetofconicsection ( <TPSetOfConicSection> )
                    |  tpsetoffillarea ( <TPSetOfFillArea> )
                    |  tpsetofellipticsector ( <TPSetOfEllipticSector> )
                    |  tpsetofellipticsegment ( <TPSetOfEllipticSegment> )
                    |  tpsetofellipticdisc ( <TPSetOfEllipticDisc> )
                    |  tpsetofclosednurb ( <TPSetOfClosedNURB> )
                    |  tpdesign ( <TPDesign> )
<TilingTran>        ::= TilingTran ( <Matrix23> )
<Timeout>           ::= <real>
<TPDesign>          ::= TPDesign ( <StencilName> × <LCStencilOrigin> × <StencilTran> ×
                            <TilingName> × <LCTilingOrigin> × <TilingTran> )
<TPSetOfClosedNURB>    ::=TPSetOfClosedNURB ( <SetOfLCNURB> × <ColrSpec> )
<TPSetOfConicSection>  ::= TPSetOfConicSection ( <SetOfLCConicSection> × <ColrSpec> )
<TPSetOfEllipticDisc>  ::= TPSetOfEllipticDisc ( <SetOfEllipticDisc> × <ColrSpec> )
<TPSetOfEllipticSector>    ::=TPSetOfEllipticSector ( <SetOfLCConicSection> × <ColrSpec> )
<TPSetOfEllipticSegment> ::=TPSetOfEllipticSegment ( <SetOfLCConicSection> × <ColrSpec> )
<TPSetOfFillArea>   ::= TPSetOfFillArea ( <SetOfLCFillArea> × <ColrSpec> )
<TPSetOfNURB>       ::= TPSetOfNURB ( <LCSetOfNURB> × <ColrSpec> )
<TPSetOfPolyline>   ::= TPSetOfPolyline ( <LCSetOfPolyline> × <ColrSpec> )
<TransMode>         ::= REPLACE | PRE | POST
<TriggerSet>        ::= PSET ( { <TriggerId> {, <TriggerId>}o }o )
<Vec>               ::= Vec ( <WCOrd> , <WCOrd> )
<VertAlign>         ::= NORMAL | TOP | CAP | HALF | BASE | BOTTOM
<ViewInd>           ::= <integer>
<ViewInd ↛ ViewRep>     ::=PSET(<ViewIndViewRep>
                            {, <ViewIndViewRep> }o )
<ViewInd1>          ::= <integer>
<ViewIndViewInd>    ::= ViewIndViewInd ( <ViewInd> → <ViewInd> )
<ViewIndViewRep>    ::= ViewIndViewRep( <ViewInd> → <ViewIndViewRep> )
<ViewPri>           ::= PSET(<ViewIndViewInd>
                            {, <ViewIndViewInd> }o )
<ViewRep>           ::= ViewRep ( <SelectCrit> × <Matrix23> × <Matrix23> × <ViewScissor> )
<ViewScissor>       ::= ViewScissor ( <ClipInd> × <LDCRectSet> × <ShieldInd> × <LDCRectSet> )
<ViewUpVec>         ::= ViewUpVec ( <WCOrd> , <WCOrd> )
<Vis>               ::= VISIBLE | INVISIBLE
<VisEffSt>          ::= SUPPRESS | ALLOW
<VpInPri>           ::= SEQ (<ViewInd> {, <ViewInd>} 63 )
<WCNDCPoint>        ::= wcpoint ( <WCPoint> ) | ndcpoint ( <NDCPoint> )
<WCOrd>             ::= <real>
<WCOrdPos>          ::= <real>
<WCPoint>           ::= WCPoint( <real> , <real> )
<WCWin>             ::= WCWin ( <WCOrd> , <WCOrd> , <WCOrd> , <WCOrd> )
<WDTEntName>        ::= PREDCURVEBUNS
                    |  WSTYPE
                    |  WSSTATUS
                    |  DCUNITS
                    |  DISPSIZE
                    |  DISPKIND
                    |  AVAILCURVETYPES
                    |  NUMAVAILCURVEWIDTHS
                    |  NOMCURVEWIDTH
                    |  MINCURVEWIDTH
```

| MAXCURVEWIDTH
| PREDCURVEBUNS
| AVAILMARKERTYPES
| NUMAVAILMARKERSIZES
| NOMMARKERSIZE
| MINMARKERSIZE
| MAXMARKERSIZE
| PREDMARKERBUNS
| AVAILINTSTYLES
| AVAILHATCHSTYLES
| PREDAREABUNS
| AVAILEDGETYPES
| NUMAVAILEDGEWIDTHS
| NOMEDGEWIDTH
| MINEDGEWIDTH
| MAXEDGEWIDTH
| PREDPATS
| AVAILFONTPRECS
| NUMAVAILCHAREXPANS
| MINCHAREXPAN
| MAXCHAREXPAN
| NUMAVAILTEXTHT
| MINTEXTHT
| MAXTEXTHT
| PREDCHARBUNS
| AVAILCOLRMODELS
| COLRCHARACTERISTICS
| NUMAVAILCOLRS
| COLRAVAIL
| PREDCOLRREPS
| AVAILGDPS
| AVAILINDEVS
| INITVALUES
| AVAILMEASUREIDS
| AVAILTRIGGERIDS
| DEVCOMPOSE

<WDTEntValue ::= predcurvebuns ( <CurveBunTab> )
| wstype ( <WsType> )
| wsstatus ( <WsStatus> )
| dcunits ( <DCUnits> )
| dispsize ( <DispSize> )
| dispkind ( <DispKind> )
| availcurvetypes ( <***P*** CurveType> )
| numavailcurvewidths ( <integer> )
| nomcurvewidth ( <DCOrd> )
| mincurvewidth ( <DCOrd> )
| maxcurvewidth ( <DCOrd> )
| predcurvebuns ( <CurveBunTab> )
| availmarkertypes ( <***P*** MarkerType> )
| numavailmarkersizes ( <integer> )
| nommarkersize ( <DCOrd> )
| minmarkersize ( <DCOrd> )