INTERNATIONAL STANDARD

**ISO/IEC 13249-5**

Second edition
2003-11-01

# Information technology — Database languages — SQL multimedia and application packages —

## Part 5:
## Still image

*Technologies de l'information — Langages de bases de données — Multimédia SQL et paquetages d'application —*

*Partie 5: Image fixe*

Reference number
ISO/IEC 13249-5:2003(E)

## Contents

## Tables Page

# 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.

International Standards are drafted in accordance with the rules given in the ISO/IEC Directives, Part 2.

The main task of the joint technical committee is to prepare International Standards. 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.

Attention is drawn to the possibility that some of the elements of this document may be the subject of patent rights. ISO and IEC shall not be held responsible for identifying any or all such patent rights.

ISO/IEC 13249-5 was prepared by Joint Technical Committee ISO/IEC JTC 1, *Information technology*, Subcommittee SC 32, *Data management and interchange*.

This second edition cancels and replaces the first edition (ISO/IEC 13249-5:2001), which has been technically revised.

ISO/IEC 13249 consists of the following parts, under the general title *Information technology — Database languages — SQL multimedia and application packages*:

— *Part 1: Framework*

— *Part 2: Full-Text*

— *Part 3: Spatial*

— *Part 5: Still image*

— *Part 6: Data mining*

# Introduction

The purpose of this International Standard is to define multimedia and application specific types and their associated routines using the user-defined features in ISO/IEC 9075.

This document is based on the content of ISO/IEC International Standard Database Language (SQL).

The organization of this part of ISO/IEC 13249 is as follows:

1) Clause 1, "Scope", specifies the scope of this part of ISO/IEC 13249.
2) Clause 2, "Normative references", identifies additional standards that, through reference in this part of ISO/IEC 13249, constitute provisions of this part of ISO/IEC 13249.
3) Clause 3, "Definitions, notations, and conventions", defines the notations and conventions used in this part of ISO/IEC 13249.
4) Clause 4, "Concepts", presents concepts used in the definition of this part of ISO/IEC 13249.
5) Clause 5, "Still Image Types", defines the still image user-defined types and associated routines.
6) Clause 6, "Feature Types", defines the user-defined types provided for the manipulation of still image features.
7) Clause 7, "SQL/MM Still Image Information Schema" defines the SQL/MM Still Image Information Schema.
8) Clause 8, "SQL/MM Still Image Definition Schema" defines the SQL/MM Still Image Definition Schema.
9) Clause 9, "Status Codes", defines the SQLSTATE codes used in this part of ISO/IEC 13249.
10) Clause 10, "Conformance", defines the criteria for conformance to this part of ISO/IEC 13249.
11) Annex A, "Implementation-defined elements", is an informative Annex. It lists those features for which the body of this part of ISO/IEC 13249 states that the syntax or meaning or effect on the database is partly or wholly implementation-defined, and describes the defining information that an implementer shall provide in each case.
12) Annex B, "Implementation-dependent elements", is an informative Annex. It lists those features for which the body of this part of ISO/IEC 13249 states explicitly that the meaning or effect on the database is implementation-dependent.

# Information technology — Database languages — SQL multimedia and application packages —

# Part 5: Still image

## 1 Scope

This part of ISO/IEC 13249:

a) introduces the still image part of ISO/IEC 13249 (all parts);

b) gives the references necessary for this part of ISO/IEC 13249;

c) defines notations and conventions specific to this part of ISO/IEC 13249;

d) defines concepts specific to this part of ISO/IEC 13249;

e) defines the still image user-defined types and their associated routines.

The still image user-defined types defined in this part of ISO/IEC 13249 adhere to the following.

— A still image user-defined type is generic to image handling. It addresses the need to store, manage and retrieve information based on aspects of inherent image characteristics such as height, width and format and based on image features such as average color, color histogram, positional color and texture. It also addresses the need to employ manipulation such as rotation, scaling, as well as similarity assessment.

— A still image user-defined type does not redefine the database language SQL directly or in combination with another still image data type.

The still image user-defined types are applicable to all different image formats. However, not all functionality can be used with all known still image formats.

An implementation of this part of ISO/IEC 13249 may exist in environments that also support information and content management, decision support, data mining and data warehousing systems.

Application areas addressed by implementations of this part of ISO/IEC 13249 include, but are not restricted to, graphics, multimedia, scientific research and medicine.

## 2 Normative references

The following referenced documents are indispensable for the application of this document. For dated references, only the edition cited applies. For undated references, the latest edition of the referenced document (including any amendments) applies.

ISO/IEC 9075-9:2001, *Information technology — Database languages — SQL — Part 9: Management of External Data (SQL/MED)*

ISO/IEC 13249-1:2002, *Information technology — Database languages — SQL multimedia and application packages — Part 1: Framework*

ISO/IEC 10918-1:1994, *Information technology — Digital compression and coding of continuous-tone still images: Requirements and guidelines*

ISO/IEC 15444-1:2000, *Information technology — JPEG 2000 image coding system — Part 1: Core coding system*

# 3 Terms and definitions, notations and conventions

## 3.1 Terms and definitions

### 3.1.1 Terms and definitions provided in ISO/IEC 13249-1:2002

This part of ISO/IEC 13249 makes use of all terms defined in ISO/IEC 13249-1:2002.

### 3.1.2 Terms and definitions provided in this part of ISO/IEC 13249

For the purposes of this document, the following terms and definitions apply.

**3.1.2.1**
**basic image feature**
a basic image feature is an image feature that is not a composite feature

**3.1.2.2**
**color space**
a set of conventions how to represent a color value

**3.1.2.3**
**composite feature**
an image feature, which consists of basic image features and their associated weights

**3.1.2.4**
**image format**
a set of conventions for storing the image data of still images in a specific compressed or uncompressed file format

**3.1.2.5**
**image feature**
characteristic (other than inherent image characteristics) of the raw image

**3.1.2.6**
**inherent image characteristics**
image format and particular physical characteristics of a still image

**3.1.2.7**
**list of weighted features**
see composite feature

**3.1.2.8**
**most significant color**
a single color representing the dominant color of a part of an image

**3.1.2.9**
**raw image**
a binary string that represents a certain image

**3.1.2.10**
**similarity of images**
a numerical measure obtainable by the comparison of two images; the measure is based on image features

**3.1.2.11**
**still image**
see "image" in Subclause 3.1.5, "Terms and definitions taken from ISO/IEC 15444-1:2000"

**3.1.2.12**
**thumbnail**
a raw image which was obtained from another raw image by downsizing

### 3.1.3 Terms and definitions taken from ISO/IEC 9075-9:2001

This part of ISO/IEC 13249 makes use of the following terms defined in ISO/IEC 9075-9:2001:

a) datalink

### 3.1.4 Terms and definitions taken from ISO/IEC 10918-1:1994

This part of ISO/IEC 13249 makes use of the following terms defined in ISO/IEC 10918-1:1994:

a) columns

NOTE 1 The use of "columns" here is as defined in the JPEG standard and not as defined in the SQL standard.

b) component

c) (digital) (still) image

NOTE 2 Parentheses around the text "digital" and "still" is a convention used by ISO/IEC 10918-1 to denote that the phrases "digital image", "still image", and "image" are interchangeable.

d) image data

e) interchange format

f) (number of) lines

NOTE 3 Parentheses around the text "number of" is a convention used by ISO/IEC 10918-1 to denote that the phases "number of lines" and "lines" are interchangeable.

g) sample

### 3.1.5 Terms and definitions taken from ISO/IEC 15444-1:2000

This part of ISO/IEC 13249 makes use of the following terms defined in ISO/IEC 15444-1:2000:

a) file format

b) reference grid

## 3.2 Notations

### 3.2.1 Notations provided in ISO/IEC 13249-1:2002

The notations used in this part of ISO/IEC 13249 are defined in ISO/IEC 13249-1:2002.

### 3.2.2 Notations provided in this part of ISO/IEC 13249

This part of ISO/IEC 13249 uses the prefix 'SI_' for user-defined types, attributes and SQL-invoked routine names.

## 3.3 Conventions

The conventions used in this part of ISO/IEC 13249 are defined in ISO/IEC 13249-1:2002.

# 4 Concepts

## 4.1 Introduction

In the context of this part of ISO/IEC 13249, still images are effectively images consisting of 2-dimensional arrays of samples. The internal representation of components and samples in the raw image itself is image format specific. The color information in the image data might be represented by an index in a color look up table, represented by multiple components, multiple binary strings which represent the color values within a single component, or in any other image format specific way.

An image format is a set of conventions for storing images in a file format. An image consists of the representation and organization of data that constitute the samples of the components, and prescriptions about auxiliary data that control the interpretation and processing of the image information according to that format.

A color space, which is used to represent the color values in the image, is either defined implicitly by the image format or described in the header information of the raw image.

An image format is referenced by format indications. A format indication is a character string whose format and content is implementation-defined.

A binary string that adheres to a certain image format is called a raw image.

The inherent image characteristics of a raw image consist of:

- the format of the raw image;
- the width of the raw image is the number of points in the horizontal dimension of the reference grid of the image data;
- the height of the raw image is the number of points in the vertical dimension of the reference grid of the image data.

An image format is a format supported by an implementation (for short: a supported format) if the implementation is able to derive the inherent image characteristics from the raw image.

This part of ISO/IEC 13249 defines types and routines with provisions for storing and manipulating still images. This part of ISO/IEC 13249 consists of the following parts:

- The data type *SI_StillImage*, a value of which has the following structure:
  - a digital representation of a still image or a reference to it;
  - format conventions used for representing that still image;
  - physical characteristics of that still image (such as its height and width).
- Methods on the data type *SI_StillImage* for:
  - constructing *SI_StillImage* values;
  - obtaining the digital representation of an image, a reference to it, or the inherent image characteristics of the image;
  - obtaining a scaled still image from an *SI_StillImage* value;
  - obtaining a resized still image from an *SI_StillImage* value;
  - obtaining a rotated still image from an *SI_StillImage* value;
  - obtaining a thumbnail from an *SI_StillImage* value;
  - deriving a metric value that characterizes the content of the image with respect to features values.

- Feature data types that abstract from certain characteristics of the pictorial information contained in images; these data types provide facilities for:
  - deriving feature values from a given image;
  - constructing feature values;
  - deriving metric values that characterize the content of images with respect to feature values.
- Information Schema views that provide data describing certain capabilities of an implementation of this part of ISO/IEC 13249.

A conforming implementation of this part of ISO/IEC 13249 shall be based on SQL-implementations that support Core SQL as defined by ISO/IEC 9075-9:2001. A number of provisions are made for that purpose:

- No function name overloading is used for SQL-invoked regular functions that are intended for public use;
- For every method that is intended for public use, a corresponding SQL-invoked regular function is specified that provides the same services as the associated method;
- The lengths of the names of schemata, types and SQL-invoked regular functions that are intended for public use do not exceed 18 characters. If the name of a view of the Still Image Information Schema exceeds 18 characters, an equivalent view with a short identifier is also specified.

## 4.2 Concepts taken from ISO/IEC 9075-9:2001

The following concepts defined in ISO/IEC 9075 are used in this part of ISO/IEC 13249.

a) binary string

b) EXECUTE privilege

c) function

d) SQL-invoked regular function

## 4.3 Types representing still images

### 4.3.1 Attributes of the SI_StillImage type

The *SI_StillImage* type is an abstraction for still images, using the following attributes:

- The attribute *SI_content* to represent the raw image;
- The attribute *SI_contentLength* to represent the length of the raw image;
- The attribute *SI_reference* to represent the reference to the image content of the raw image;
- The attribute *SI_format* to represent a format indication; it identifies the image format of the raw image;
- The attribute *SI_width* to represent the width of the raw image;
- The attribute *SI_height* to represent the height of the raw image.
- The attribute *SI_retainFeatures* to specify that the image's features are to be computed and maintained automatically by the implementation.
- Additional implementation-dependent attributes to store the image features or a representation of image features.

### 4.3.2 Methods of the SI_StillImage type

The type *SI_StillImage* provides the following methods for public use:

- *SI_StillImage*: constructs an *SI_StillImage* value from a raw image;
- *SI_StillImage*: constructs an *SI_StillImage* value from a raw image and a character string representing a format indication; this method allows for user-supplied format indication when *SI_StillImage* values are to be constructed from a raw image whose format is not a supported one;

- *SI_StillImage*: constructs an *SI_StillImage* value from a reference to a raw image;
- *SI_StillImage*: constructs an *SI_StillImage* value from a reference to a raw image and a character string representing a format indication; this method allows for user-supplied format indication when *SI_StillImage* values are to be constructed from a raw image whose format is not a supported one;
- *SI_StillImage*: constructs an *SI_StillImage* value from a raw image and also extracts the supported image features. This method allows for a user-supplied integer value to indicate the need for image feature extraction. If the user-supplied integer value is 1 (one), the image features are extracted during value construction. If the user supplied integer value is 0 (zero), the method has the same effect as the invocation of the constructor method *SI_StillImage(content)*;
- *SI_setContent*: has the same effect as the invocation of the mutator function for the *SI_content* attribute, but additionally adjusts the values of the attributes that represent the inherent image characteristics;
- *SI_changeFormat*: has the same effect as the invocation of the mutator function for the *SI_format* attribute, but additionally adjusts the value of the attribute *SI_content* and the values of the attributes which represent the inherent image characteristics. The format conversion fails if the conversion between the source image format and the target image format is not supported;
- *SI_Scale*: obtains a scaled image from an *SI_StillImage* value. The raw image is scaled to fit into a bounding box defined by the specified height and width. The aspect ratio of the image's height and width shall be maintained;
- *SI_Scale*: obtains a scaled image from an *SI_StillImage* value. The raw image is scaled by the specified factor. The aspect ratio of the image’s height and width shall be maintained;
- *SI_Resize*: obtains a resized image from an *SI_StillImage* value. The raw image is resized to the specified height and width. The aspect ratio of the image’s height and width does not have to be maintained;
- *SI_Rotate*: obtains a rotated image from an *SI_StillImage* value. The raw image is rotated by the defined angle. For rotations by an angle that is not a multiple of 90 degrees, the raw image is padded and the height and width of the resulting still image are adjusted;
- *SI_Thumbnail*: obtains a thumbnail from an *SI_StillImage* value;
- *SI_InitFeatures*: indicates that the attributes for supported features (average color, color histogram, positional color, and texture) of the image are to be computed and kept updated.
- *SI_ClearFeatures*: indicates that the attributes for supported features of the image are to be discarded and no longer updated.
- *SI_Score:* returns a numerical value that is a measure of the similarity of the image feature value to a given feature value;
- *SI_content*: returns the representation of the raw image;
- *SI_contentLength*: returns the length in bytes of the representation of the raw image;
- *SI_reference*: returns the reference to the image content of the raw image;
- *SI_format*: returns the format indication of the image;
- *SI_height*: returns the number of points in the vertical dimension of the reference grid of the image;
- *SI_width*: returns the number of points in the horizontal dimension of the reference grid of the image.

## 4.4 Image features

Image features (for short: features) are used to characterize the pictorial information of an image by means other than inherent image characteristics. This part of ISO/IEC 13249 supports four basic features and one composite feature. The basic features are:

- *Average color feature*: this feature characterizes an image by its average color;
- *Color histogram feature*: this feature characterizes an image by the relative frequencies of the colors exhibited by the raw image;

- *Positional color feature*: let an image be divided into *n* by *m* rectangles; the positional color feature characterizes that image by the *n* by *m* most significant colors of these rectangles;
- *Texture feature*: this feature characterizes an image by the size of repeating items (coarseness), brightness variations (contrast), and the predominant direction (directionality).

All basic features can be derived from images. In addition, the two basic features average color feature and color histogram feature can be constructed by means of numerical values.

The composite feature is a list of up to 4 basic features, each of a different feature type. All the basic features in the composite feature are associated with a feature weight.

Features are represented by feature types. Values of those types are used for obtaining a quantitative measure for the similarity between two images represented as *SI_StillImage* values, say, $I_1$ and $I_2$. If $F_2$ is some feature that characterizes the image $I_2$, then a similarity measure for the images $I_1$ and $I_2$ is obtained from $F_2$ by a method *SI_Score* that takes $I_1$ as its parameter. For a given pair of images, the obtained similarity depends on the kind of feature used for comparison; the exact relationship is implementation-dependent.

### 4.4.1 Feature types

The basic features are represented by the following feature types:

| | |
|---|---|
| - Average color feature: | *SI_AverageColor*; |
| - Color histogram feature: | *SI_ColorHistogram*; |
| - Positional color feature: | *SI_PositionalColor*; |
| - Texture feature: | *SI_Texture*. |

The composite feature is represented by the feature type *SI_FeatureList*.

For all basic features methods and functions are provided that derive the corresponding feature value from an *SI_StillImage* value. The functions are:

| | |
|---|---|
| - Average color feature: | *SI_findAvgClr*; |
| - Color histogram feature: | *SI_findClrHstgr*; |
| - Positional color feature: | *SI_findPstnlClr*; |
| - Texture feature: | *SI_findTexture*. |

*SI_AverageColor* and *SI_ColorHistogram* values can also be obtained by methods *SI_AverageColor* and *SI_ColorHistogram*, respectively. The parameter of the first method is a color value that is used to represent the intended average color. The second method takes a first color value and a first frequency and returns an initial color histogram. This initial color histogram can be extended using the method *SI_Append*.

*SI_FeatureList* values shall be constructed from basic feature values and associated weights.

The observer and mutator functions of the basic feature types are not intended for public use. Thus, there are no GRANT statements granting EXECUTE privilege on these functions.

### 4.4.2 Assessing the similarity of images

Every feature type has a method *SI_Score*. This method can be used for obtaining numerical values that measure the similarity between two images. To that end, a distance function is used for this measurement. The returned numerical value for the distance indicates the difference between a given feature value and a still image value. Let $F_1$ be some feature value that characterizes an *SI_StillImage* values $I_1$. Then

$F_1$`.SI_Score(`$I_2$`)`

returns a measure for the similarity of the *SI_StillImage* value $I_2$ to the feature value $F_1$.

## 4.5 Complementary SQL-invoked regular functions

To ease conformance for implementation of this part of ISO/IEC 13249, each method intended for public use is complemented by an SQL-invoked regular function.

For each such method, the type of specified method, the method name, parameter types (if any), and the name of the corresponding SQL-invoked regular function is listed in Table 1 — Method and function name correspondences. Since the names of these functions are unique, their parameter types are not given.

**Table 1 — Method and function name correspondences**

| Type Name | Method Name | Parameter Types (if any) | Function Name |
|---|---|---|---|
| SI_StillImage | SI_StillImage | BINARY LARGE OBJECT | SI_mkStillImage1 |
| SI_StillImage | SI_StillImage | BINARY LARGE OBJECT, CHARACTER VARYING | SI_mkStillImage2 |
| SI_StillImage | SI_StillImage | DATALINK | SI_mkStillImage3 |
| SI_StillImage | SI_StillImage | DATALINK, CHARACTER VARYING | SI_mkStillImage4 |
| SI_StillImage | SI_StillImage | BINARY LARGE OBJECT, INTEGER | SI_mkStillImage5 |
| SI_StillImage | SI_StillImage | BINARY LARGE OBJECT, CHARACTER VARYING, INTEGER | SI_mkStillImage6 |
| SI_StillImage | SI_setContent | BINARY LARGE OBJECT | SI_chgContent |
| SI_StillImage | SI_changeFormat | CHARACTER VARYING | SI_convertFormat |
| SI_StillImage | SI_Scale | INTEGER, INTEGER | SI_scaleImage |
| SI_StillImage | SI_Scale | DOUBLE PRECISION | SI_zoomImage |
| SI_StillImage | SI_Resize | INTEGER, INTEGER | SI_resizeImage |
| SI_StillImage | SI_Rotate | DOUBLE PRECISION | SI_rotateImage |
| SI_StillImage | SI_Thumbnail | | SI_getThmbnl |
| SI_StillImage | SI_Thumbnail | INTEGER, INTEGER | SI_getSizedThmbnl |
| SI_StillImage | SI_InitFeatures | | SI_setImageFtrs |
| SI_StillImage | SI_ClearFeatures | | SI_resetImageFtrs |
| SI_StillImage | SI_content | | SI_getContent |
| SI_StillImage | SI_contentLength | | SI_getContentLngth |
| SI_StillImage | SI_reference | | SI_getReference |
| SI_StillImage | SI_format | | SI_getFormat |
| SI_StillImage | SI_height | | SI_getHeight |
| SI_StillImage | SI_width | | SI_getWidth |
| SI_AverageColor | SI_AverageColor | SI_StillImage | SI_findAvgClr |
| SI_AverageColor | SI_AverageColor | SI_Color | SI_mkAvgClr |
| SI_AverageColor | SI_Score | SI_StillImage | SI_ScoreByAvgClr |
| SI_ColorHistogram | SI_ColorHistogram | SI_StillImage | SI_findClrHstgr |

**Table 1** (*continued*)

| Type Name | Method Name | Parameter Types (if any) | Function Name |
|---|---|---|---|
| SI_ColorHistogram | SI_ColorHistogram | SI_Color,<br>DOUBLE PRECISION | SI_mkClrHstgr |
| SI_ColorHistogram | SI_ColorHistogram | SI_Color ARRAY,<br>DOUBLE PRECISION ARRAY | SI_arrayClrHstgr |
| SI_ColorHistogram | SI_Append | SI_Color,<br>DOUBLE PRECISION | SI_appendClrHstgr |
| SI_ColorHistogram | SI_Score | SI_StillImage | SI_ScoreByClrHstgr |
| SI_PositionalColor | SI_PositionalColor | SI_StillImage | SI_findPstnlClr |
| SI_PositionalColor | SI_Score | SI_StillImage | SI_ScoreByPstnlClr |
| SI_Texture | SI_Texture | SI_StillImage | SI_findTexture |
| SI_Texture | SI_Score | SI_StillImage | SI_ScoreByTexture |
| SI_FeatureList | SI_FeatureList | SI_AverageColor,<br>DOUBLE PRECISION,<br>SI_ColorHistogram,<br>DOUBLE PRECISION,<br>SI_PositionalColor,<br>DOUBLE PRECISION,<br>SI_Texture,<br>DOUBLE PRECISION | SI_mkFtrList |
| SI_FeatureList | SI_setFeature | SI_AverageColor,<br>DOUBLE PRECISION | SI_setAvgClrFtr |
| SI_FeatureList | SI_setFeature | SI_ColorHistogram,<br>DOUBLE PRECISION | SI_setClrHstgrFtr |
| SI_FeatureList | SI_setFeature | SI_PositionalColor,<br>DOUBLE PRECISION | SI_setPstnlClrFtr |
| SI_FeatureList | SI_setFeature | SI_Texture,<br>DOUBLE PRECISION | SI_setTextureFtr |
| SI_FeatureList | SI_AvgClrFtr | | SI_getAvgClrFtr |
| SI_FeatureList | SI_AvgClrFtrWght | | SI_getAvgClrFtrW |
| SI_FeatureList | SI_ClrHstgrFtr | | SI_getClrHstgrFtr |
| SI_FeatureList | SI_ClrHstgrFtrWght | | SI_getClrHstgrFtrW |
| SI_FeatureList | SI_PstnlClrFtr | | SI_getPstnlClrFtr |
| SI_FeatureList | SI_PstnlClrFtrWght | | SI_getPstnlClrFtrW |
| SI_FeatureList | SI_TextureFtr | | SI_getTextureFtr |
| SI_FeatureList | SI_TextureFtrWght | | SI_getTextureFtrW |
| SI_FeatureList | SI_Score | SI_StillImage | SI_ScoreByFtrList |
| SI_Color | SI_RGBColor | INTEGER,<br>INTEGER,<br>INTEGER | SI_mkRGBClr |

## 4.6 Auxiliary type SI_Color

Color values are encapsulated by the type *SI_Color*. An implementation shall provide constructor methods to obtain *SI_Color* values. Each constructor method is provided for a specific color space. Each function takes parameters that represent the intended color value in this color space. A constructor method *SI_RGBColor*, for the RGB color space, is defined in this part of ISO/IEC 13249. Constructor methods for other color spaces are implementation-defined.

## 4.7 The Still Image Information Schema

This part of ISO/IEC 13249 prescribes an Information Schema called SI_INFORMTN_SCHEMA. It contains views for the following purposes:

- a view SI_FEATURES, which lists the feature indications of the supported features;
- a view SI_IMAGE_FORMATS, which lists the format indications of the supported image formats and which operations are supported for images with this format;
- a view SI_IMAGE_FORMAT_CONVERSIONS, which lists pairs of format indications of image formats for which format conversions are supported;
- a view SI_IMAGE_FORMAT_FEATURES, which lists pairs of format indications and feature indications; images of the indicated image format support the extraction of the indicated feature and scoring with respect to that feature;
- a view SI_THUMBNAIL_FORMATS, which lists the format indications of image formats from which thumbnails can be derived;
- a view SI_VALUES, which lists implementation-defined meta-variable and their values.

# 5 Still Image Types

The types in this family provide for the storage and retrieval of still image values.

## 5.1 SI_StillImage Types and Routines

### 5.1.1 SI_StillImage Type

**Purpose**

The *SI_StillImage* type provides the definition of a still image type.

**Definition**

```
CREATE TYPE SI_StillImage
   AS (
      SI_content BINARY LARGE OBJECT(SI_MaxContentLength),
      SI_contentLength INTEGER,
      SI_reference DATALINK SI_DLLinkControl SI_DLIntegrityOption
         SI_DLReadPermission SI_DLWritePermission
         SI_DLRecoveryOption SI_DLUnlinkOption,
      SI_format CHARACTER VARYING(SI_MaxFormatLength),
      SI_height INTEGER,
      SI_width INTEGER,
      SI_retainFeatures SMALLINT DEFAULT 0
      -- Additional implementation-dependent attributes
      -- to store computed image features
   )
   INSTANTIABLE
   NOT FINAL

   CONSTRUCTOR METHOD SI_StillImage
      (content BINARY LARGE OBJECT(SI_MaxContentLength))
      RETURNS SI_StillImage
      SELF AS RESULT
      LANGUAGE SQL
      DETERMINISTIC
      CONTAINS SQL
      RETURNS NULL ON NULL INPUT,

   CONSTRUCTOR METHOD SI_StillImage
      (content BINARY LARGE OBJECT(SI_MaxContentLength),
       explicitFormat CHARACTER VARYING(SI_MaxFormatLength))
      RETURNS SI_StillImage
      SELF AS RESULT
      LANGUAGE SQL
      DETERMINISTIC
      READS SQL DATA
      RETURNS NULL ON NULL INPUT,

   CONSTRUCTOR METHOD SI_StillImage
      (reference DATALINK)
      RETURNS SI_StillImage
      SELF AS RESULT
      LANGUAGE SQL
      DETERMINISTIC
      CONTAINS SQL
      RETURNS NULL ON NULL INPUT,
CONSTRUCTOR METHOD SI_StillImage
   (reference DATALINK,
    explicitFormat CHARACTER VARYING(SI_MaxFormatLength))
   RETURNS SI_StillImage
   SELF AS RESULT
   LANGUAGE SQL
   DETERMINISTIC
   READS SQL DATA
   RETURNS NULL ON NULL INPUT,

CONSTRUCTOR METHOD SI_StillImage
   (content BINARY LARGE OBJECT(SI_MaxContentLength),
    maintainFeatures INTEGER)
   RETURNS SI_StillImage
   SELF AS RESULT
   LANGUAGE SQL
   DETERMINISTIC
   CONTAINS SQL
   RETURNS NULL ON NULL INPUT,

CONSTRUCTOR METHOD SI_StillImage
   (content BINARY LARGE OBJECT(SI_MaxContentLength),
    explicitFormat CHARACTER VARYING(SI_MaxFormatLength),
    maintainFeatures INTEGER)
   RETURNS SI_StillImage
   SELF AS RESULT
   LANGUAGE SQL
   DETERMINISTIC
   READS SQL DATA
   RETURNS NULL ON NULL INPUT,

METHOD SI_setContent
   (content BINARY LARGE OBJECT(SI_MaxContentLength))
   RETURNS SI_StillImage
   SELF AS RESULT
   LANGUAGE SQL
   DETERMINISTIC
   CONTAINS SQL
   RETURNS NULL ON NULL INPUT,

METHOD SI_changeFormat
   (targetFormat CHARACTER VARYING(SI_MaxFormatLength))
   RETURNS SI_StillImage
   SELF AS RESULT
   LANGUAGE SQL
   DETERMINISTIC
   READS SQL DATA
   RETURNS NULL ON NULL INPUT,

METHOD SI_Scale
   (height INTEGER,
    width INTEGER)
   RETURNS SI_StillImage
   SELF AS RESULT
   LANGUAGE SQL
   DETERMINISTIC
   READS SQL DATA
   CALLED ON NULL INPUT,
METHOD SI_Scale
    (scaleFactor DOUBLE PRECISION)
   RETURNS SI_StillImage
   SELF AS RESULT
   LANGUAGE SQL
   DETERMINISTIC
   READS SQL DATA
   RETURNS NULL ON NULL INPUT,

METHOD SI_Resize
    (height INTEGER,
     width INTEGER)
   RETURNS SI_StillImage
   SELF AS RESULT
   LANGUAGE SQL
   DETERMINISTIC
   READS SQL DATA
   CALLED ON NULL INPUT,

METHOD SI_Rotate
    (angle DOUBLE PRECISION)
   RETURNS SI_StillImage
   SELF AS RESULT
   LANGUAGE SQL
   DETERMINISTIC
   READS SQL DATA
   RETURNS NULL ON NULL INPUT,

METHOD SI_Thumbnail()
   RETURNS SI_StillImage
   SELF AS RESULT
   LANGUAGE SQL
   DETERMINISTIC
   READS SQL DATA
   RETURNS NULL ON NULL INPUT,

METHOD SI_Thumbnail
    (height INTEGER,
     width INTEGER)
   RETURNS SI_StillImage
   SELF AS RESULT
   LANGUAGE SQL
   DETERMINISTIC
   READS SQL DATA
   RETURNS NULL ON NULL INPUT,

METHOD SI_InitFeatures()
   RETURNS SI_StillImage
   SELF AS RESULT
   LANGUAGE SQL
   DETERMINISTIC
   READS SQL DATA
   RETURNS NULL ON NULL INPUT,

METHOD SI_ClearFeatures()
   RETURNS SI_StillImage
   SELF AS RESULT
   LANGUAGE SQL
   DETERMINISTIC
   READS SQL DATA
   RETURNS NULL ON NULL INPUT,
    METHOD SI_Score
        (averageColor SI_AverageColor)
       RETURNS DOUBLE PRECISION
       LANGUAGE SQL
       DETERMINISTIC
       READS SQL DATA
       RETURNS NULL ON NULL INPUT,

    METHOD SI_Score
        (colorHistogram SI_ColorHistogram)
       RETURNS DOUBLE PRECISION
       LANGUAGE SQL
       DETERMINISTIC
       READS SQL DATA
       RETURNS NULL ON NULL INPUT,

    METHOD SI_Score
        (positionalColor SI_PositionalColor)
       RETURNS DOUBLE PRECISION
       LANGUAGE SQL
       DETERMINISTIC
       READS SQL DATA
       RETURNS NULL ON NULL INPUT,

    METHOD SI_Score
        (texture SI_Texture)
       RETURNS DOUBLE PRECISION
       LANGUAGE SQL
       DETERMINISTIC
       READS SQL DATA
       RETURNS NULL ON NULL INPUT,

    METHOD SI_Score
        (featureList SI_FeatureList)
       RETURNS DOUBLE PRECISION
       LANGUAGE SQL
       DETERMINISTIC
       READS SQL DATA
       RETURNS NULL ON NULL INPUT
```

**Definitional Rules**

1) *SI_MaxContentLength* is the implementation-defined maximum length for the binary representation of the *SI_StillImage* attribute *SI_content*.
2) *SI_MaxFormatLength* is the implementation-defined maximum length for the character representation of an image format indication.
3) *SI_DLLinkControl* is the implementation-defined <datalink control definition> for the attribute *SI_reference* of the *SI_StillImage* type.
4) *SI_DLIntegrityOption* is the implementation-defined <integrity option> for the attribute *SI_reference* of the *SI_StillImage* type.
5) *SI_DLReadPermission* is the implementation-defined <read permission> for the attribute *SI_reference* of the *SI_StillImage* type.
6) *SI_DLWritePermission* is the implementation-defined <write permission> for the attribute *SI_reference* of the *SI_StillImage* type.
7) *SI_DLRecoveryOption* is the implementation-defined <recovery option> for the attribute *SI_reference* of the *SI_StillImage* type.

8) *SI_DLUnlinkOption* is the implementation-defined <unlink option> for the attribute *SI_reference* of the *SI_StillImage* type.

9) The attributes *SI_content, SI_contentLength, SI_reference, SI_format, SI_height, SI_width*, and *SI_retainFeatures* are read-only. There are no GRANT statements granting EXECUTE privilege on the mutator functions for *SI_content, SI_contentLength, SI_reference, SI_format, SI_height*, *SI_width*, and *SI_retainFeatures*.

**Description**

1) The *SI_StillImage* type provides for public use:

   a) a method *SI_StillImage(BINARY LARGE OBJECT)*,

   b) a method *SI_StillImage(BINARY LARGE OBJECT, CHARACTER VARYING),*

   c) a method *SI_StillImage(DATALINK)*,

   d) a method *SI_StillImage(DATALINK, CHARACTER VARYING)*,

   e) a method *SI_StillImage(BINARY LARGE OBJECT, INTEGER)*,

   f) a method *SI_StillImage(BINARY LARGE OBJECT, CHARACTER VARYING, INTEGER),*

   g) a method *SI_setContent(BINARY LARGE OBJECT)*,

   h) a method *SI_changeFormat(CHARACTER VARYING)*,

   i) a method *SI_Scale(INTEGER, INTEGER)*,

   j) a method *SI_Scale(DOUBLE PRECISION)*,

   k) a method *SI_Resize(INTEGER, INTEGER)*,

   l) a method *SI_Rotate(DOUBLE PRECISION)*,

   m) a method *SI_Thumbnail()*,

   n) a method *SI_Thumbnail(INTEGER, INTEGER)*,

   o) a method *SI_InitFeatures()*,

   p) a method *SI_ClearFeatures()*,

   q) a method *SI_Score(SI_AverageColor)*,

   r) a method *SI_Score(SI_ColorHistogram)*,

   s) a method *SI_Score(SI_PositionalColor)*,

   t) a method *SI_Score(SI_Texture)*,

   u) a method *SI_Score(SI_FeatureList)*,

   v) a function *SI_mkStillImage1(BINARY LARGE OBJECT)*,

   w) a function *SI_mkStillImage2(BINARY LARGE OBJECT, CHARACTER VARYING)*,

   x) a function *SI_mkStillImage3(DATALINK)*,

   y) a function *SI_mkStillImage4(DATALINK, CHARACTER VARYING)*,

   z) a function *SI_mkStillImage5(BINARY LARGE OBJECT, INTEGER)*,

   aa) a function *SI_mkStillImage6(BINARY LARGE OBJECT, CHARACTER VARYING, INTEGER)*,

   ab) a function *SI_chgContent(SI_StillImage, BINARY LARGE OBJECT)*,

   ac) a function *SI_convertFormat(SI_StillImage, CHARACTER VARYING)*,

   ad) a function *SI_scaleImage(SI_StillImage, INTEGER, INTEGER)*,

   ae) a function *SI_zoomImage(SI_StillImage, DOUBLE PRECISION)*,

   af) a function *SI_resizeImage(SI_StillImage, INTEGER, INTEGER)*,

   ag) a function *SI_rotateImage(SI_StillImage, INTEGER, INTEGER)*, ah) a function *SI_getThmbnl(SI_StillImage)*,

ai) a function *SI_getSizedThmbnl(SI_StillImage, INTEGER, INTEGER),*

aj) a function *SI_setImageFtrs(SI_StillImage),*

ak) a function *SI_resetImageFtrs(SI_StillImage)*.

2) *SI_retainFeatures* is either 0 (zero) or 1 (one). If the value is 0 (zero), then the implementation does not compute and maintain the values of the implementation-dependent attributes representing the image features (average color, color histogram, positional color and texture); otherwise, the implementation does compute and maintain the values of those attributes. The default value is 0 (zero).

3) Implementation-dependent attributes representing the average color feature; if not the null value, contains values that represent the average color feature of the *SI_StillImage* value.

4) Implementation-dependent attributes representing the color histogram feature; if not the null value, contains values that represent the color histogram feature of the *SI_StillImage* value.

5) Implementation-dependent attributes representing the positional color feature; if not the null value, contains values that represent the positional color feature of the *SI_StillImage* value.

6) Implementation-dependent attributes representing the texture feature; if not the null value, contains values that represent the texture feature of the *SI_StillImage* value.

### 5.1.2 SI_StillImage Methods

**Purpose**

Return a specified *SI_StillImage* value.

**Definition**

```
CREATE CONSTRUCTOR METHOD SI_StillImage
    (content BINARY LARGE OBJECT(SI_MaxContentLength))
    RETURNS SI_StillImage
    FOR SI_StillImage
    BEGIN
        DECLARE fmt CHARACTER VARYING(SI_MaxFormatLength);
        DECLARE FormatError CONDITION FOR SQLSTATE '2FF10';

        SET fmt = SI_format(content);
        IF fmt IS NULL THEN
            SIGNAL FormatError
                SET MESSAGE_TEXT = 'illegal image format specification';
        END IF;
        RETURN SELF.
            SI_content(content).
            SI_contentLength(LENGTH(content)).
            SI_reference(CAST (NULL AS DATALINK)).
            SI_format(fmt).
            SI_height(SI_height(content)).
            SI_width(SI_width(content));
    END

CREATE CONSTRUCTOR METHOD SI_StillImage
    (content BINARY LARGE OBJECT(SI_MaxContentLength),
     explicitFormat CHARACTER VARYING(SI_MaxFormatLength))
    RETURNS SI_StillImage
    FOR SI_StillImage
    BEGIN
        DECLARE FormatError CONDITION FOR SQLSTATE '2FF10';
        DECLARE iFormat CHARACTER VARYING(SI_MaxFormatLength);
        DECLARE eFormat CHARACTER VARYING(SI_MaxFormatLength);
```

```
         SET iFormat = TRIM(BOTH ' ' FROM SI_format(content));
         SET eFormat = TRIM(BOTH ' ' FROM explicitFormat);
         IF eFormat IS NULL OR
            NOT (
               (SI_supportedFormat(eFormat) = 1 AND
                CASE
                  WHEN iFormat IS NULL THEN
                     FALSE
                  ELSE
                     iFormat = eFormat
                END) OR
               (SI_supportedFormat(eFormat) = 0 AND
               iFormat IS NULL)
            ) THEN
            SIGNAL FormatError
               SET MESSAGE_TEXT = 'illegal image format specification';
         END IF;
         RETURN SELF.
            SI_content(content).
            SI_contentLength(LENGTH(content)).
            SI_reference(CAST (NULL AS DATALINK)).
            SI_format(explicitFormat).
            SI_height(SI_height(content)).
            SI_width(SI_width(content));
      END

   CREATE CONSTRUCTOR METHOD SI_StillImage
       (reference DATALINK)
      RETURNS SI_StillImage
      FOR SI_StillImage
      RETURN NEW SI_StillImage(SI_getDLContent(reference)).
         SI_content(CAST (NULL AS BINARY LARGE OBJECT(SI_MaxContentLength))).
         SI_contentLength(CAST (NULL AS INTEGER)).
         SI_reference(reference)

   CREATE CONSTRUCTOR METHOD SI_StillImage
       (reference DATALINK,
        explicitFormat CHARACTER VARYING(SI_MaxFormatLength))
      RETURNS SI_StillImage
      FOR SI_StillImage
      RETURN NEW SI_StillImage(SI_getDLContent(reference),explicitFormat).
         SI_content(CAST (NULL AS BINARY LARGE OBJECT(SI_MaxContentLength))).
         SI_contentLength(CAST (NULL AS INTEGER)).
         SI_reference(reference)

   CREATE CONSTRUCTOR METHOD SI_StillImage
       (content BINARY LARGE OBJECT(SI_MaxContentLength),
        maintainFeatures INTEGER)
      RETURNS SI_StillImage
      FOR SI_StillImage
      BEGIN
         DECLARE fmt CHARACTER VARYING(SI_MaxFormatLength);
         DECLARE FormatError CONDITION FOR SQLSTATE '2FF10';

         SET fmt = SI_format(content);
         IF fmt IS NULL THEN
            SIGNAL FormatError
               SET MESSAGE_TEXT = 'illegal image format specification';
         END IF;
         RETURN
            CASE
               WHEN maintainFeatures = 1 THEN
```

```
                SELF.
                   SI_content(content).
                   SI_contentLength(LENGTH(content)).
                   SI_reference(CAST (NULL AS DATALINK)).
                   SI_format(fmt).
                   SI_height(SI_height(content)).
                   SI_width(SI_width(content)).
                   SI_InitFeatures()
             ELSE
                SELF.
                   SI_content(content).
                   SI_contentLength(LENGTH(content)).
                   SI_reference(CAST (NULL AS DATALINK)).
                   SI_format(fmt).
                   SI_height(SI_height(content)).
                   SI_width(SI_width(content))
          END;
    END

CREATE CONSTRUCTOR METHOD SI_StillImage
    (content BINARY LARGE OBJECT(SI_MaxContentLength),
     explicitFormat CHARACTER VARYING(SI_MaxFormatLength),
     maintainFeatures INTEGER)
    RETURNS SI_StillImage
    FOR SI_StillImage
    BEGIN
       DECLARE FormatError CONDITION FOR SQLSTATE '2FF10';
       DECLARE iFormat CHARACTER VARYING(SI_MaxFormatLength);
       DECLARE eFormat CHARACTER VARYING(SI_MaxFormatLength);

       SET iFormat = TRIM(BOTH ' ' FROM SI_format(content));
       SET eFormat = TRIM(BOTH ' ' FROM explicitFormat);
       IF eFormat IS NULL OR
          NOT (
             (SI_supportedFormat(eFormat) = 1 AND
              CASE
                WHEN iFormat IS NULL THEN
                   FALSE
                ELSE
                   iFormat = eFormat
              END) OR
             (SI_supportedFormat(eFormat) = 0 AND
             iFormat IS NULL)
          ) THEN
          SIGNAL FormatError
             SET MESSAGE_TEXT = 'illegal image format specification';
       END IF;
       RETURN
          CASE
             WHEN maintainFeatures = 1 THEN
                SELF.
                   SI_content(content).
                   SI_contentLength(LENGTH(content)).
                   SI_reference(CAST (NULL AS DATALINK)).
                   SI_format(explicitFormat).
                   SI_height(SI_height(content)).
                   SI_width(SI_width(content)).
                   SI_InitFeatures()
             ELSE
                SELF.
```

```
                              SI_content(content).
                              SI_contentLength(LENGTH(content)).
                              SI_reference(CAST (NULL AS DATALINK)).
                              SI_format(explicitFormat).
                              SI_height(SI_height(content)).
                              SI_width(SI_width(content))
                   END;
             END
```

**Definitional Rules**

1) *SI_MaxContentLength* is the implementation-defined maximum length for the binary representation of the *SI_StillImage* attribute *SI_content*.

2) *SI_MaxFormatLength* is the implementation-defined maximum length for the character representation of an image format indication.

**Description**

1) The method *SI_StillImage(BINARY LARGE OBJECT)* takes the following input parameter:
   a) a BINARY LARGE OBJECT value *content*.

2) For the method *SI_StillImage(BINARY LARGE OBJECT)*:
   a) Let *FMT* be the result of the value expression: *SI_Format(content)*.
   b) Case:
      i) If *FMT* is the null value, then an exception condition is raised: *SQL/MM Still Image exception – illegal image format specification*.
      ii) Otherwise, return an *SI_StillImage* value with:
         1) the *SI_content* attribute set to *content*.
         2) the *SI_contentLength* attribute set to the value expression: LENGTH(*content*).
         3) the *SI_reference* attribute set to the null value.
         4) the *SI_format* attribute set to *FMT*.
         5) the *SI_height* attribute set to the value expression: *SI_height(content)*.
         6) the *SI_width* attribute set to the value expression: *SI_width(content)*.

3) The method *SI_StillImage(BINARY LARGE OBJECT, CHARACTER VARYING)* takes the following input parameters:
   a) a BINARY LARGE OBJECT value *content*,
   b) a CHARACTER VARYING value *explicitFormat*.

4) For the method *SI_StillImage(BINARY LARGE OBJECT, CHARACTER VARYING)*:

   Case:
   a) If any of the following is True for an invocation of the method SI_StillImage(BINARY LARGE OBJECT, CHARACTER VARYING), then an exception condition is raised: SQL/MM Still Image exception – illegal image format specification.
      i) *explicitFormat* is the null value.
      ii) Both of the following are *False*:
         1) *explicitFormat* indicates a supported image format, the image format derived from *content* is not the null value, and *explicitFormat* is equivalent to the format derived from *content*.
         2) *explicitFormat* indicates an unsupported image format, and the image format derived from *content* is the null value (i.e. no supported image format can be derived from *content*).
   b) Otherwise, return an *SI_StillImage* value with:
      i) the *SI_content* attribute set to *content*.

ii) the *SI_contentLength* attribute set to the value expression: LENGTH(*content*).

iii) the *SI_reference* attribute set to the null value.

iv) the *SI_format* attribute set to *explicitFormat*.

v) the *SI_height* attribute set to the value expression: *SI_height(content)*.

vi) the *SI_width* attribute set to the value expression: *SI_width(content)*.

5) The method *SI_StillImage(DATALINK)* takes the following input parameter:

a) a DATALINK value *reference*.

6) The method *SI_StillImage(DATALINK, CHARACTER VARYING)* takes the following input parameter:

a) a DATALINK value *reference*,

b) a CHARACTER VARYING value *explicitFormat*.

7) If any of the following is _True_ for an invocation of the method *SI_StillImage(DATALINK, CHARACTER VARYING)*, then an exception condition is raised: *SQL/MM Still Image exception – illegal image format specification*.

a) *explicitFormat* is the null value.

b) Both of the following is _False_:

i) *explicitFormat* indicates a supported image format, and *explicitFormat* is equivalent to the format derived from the image content *reference* is referencing.

ii) *explicitFormat* indicates an unsupported image format, and the image format derived from the image content *reference* is referencing is the null value (i.e. no supported image format can be derived from the image content *reference* is referencing).

8) The method *SI_StillImage(BINARY LARGE OBJECT, INTEGER)* takes the following input parameter:

a) a BINARY LARGE OBJECT value *content*,

b) an INTEGER value maintainFeatures.

9) For the method *SI_StillImage(BINARY LARGE OBJECT, INTEGER)*:

a) Let *FMT* be the result of the value expression: *SI_Format(content)*.

b) Case:

i) If *FMT* is the null value, then an exception condition is raised: *SQL/MM Still Image exception – illegal image format specification*.

ii) Otherwise, return an *SI_StillImage* value with:

1) the *SI_content* attribute set to *content*.

2) the *SI_contentLength* set to the value expression: LENGTH(*content*).

3) the *SI_reference* attribute set to the null value.

4) the *SI_format* attribute set to *FMT*.

5) the *SI_height* attribute set to the value expression: *SI_height(content)*.

6) the *SI_width* attribute set to the value expression: *SI_width(content)*.

7) If *maintainFeatures* is equal 1 (one), then the *SI_StillImage* value regenerates the values of the all the implementation-dependent image feature attributes values.

10) The method *SI_StillImage(BINARY LARGE OBJECT, CHARACTER VARYING, INTEGER)* takes the following input parameters:

a) a BINARY LARGE OBJECT value *content*,

b) a CHARACTER VARYING value *explicitFormat*,

c) an INTEGER value maintainFeatures.

11) For the method *SI_StillImage(BINARY LARGE OBJECT, CHARACTER VARYING)*:

    Case:

    a) If any of the following is True for an invocation of the method SI_StillImage(BINARY LARGE OBJECT, CHARACTER VARYING, *INTEGER*), then an exception condition is raised: SQL/MM Still Image exception – illegal image format specification.

        i) *explicitFormat* is the null value.

        ii) Both of the following are *False*:

            1) *explicitFormat* indicates a supported image format, the image format derived from *content* is not the null value, and *explicitFormat* is equivalent to the format derived from *content*.

            2) *explicitFormat* indicates an unsupported image format, and the image format derived from *content* is the null value (i.e. no supported image format can be derived from *content*).

    b) Otherwise, return an *SI_StillImage* value with:

        i) the *SI_content* attribute set to *content*.

        ii) the *SI_contentLength* attribute set to the value expression: LENGTH(*content*).

        iii) the *SI_reference* attribute set to the null value.

        iv) the *SI_format* attribute set to *explicitFormat*.

        v) the *SI_height* attribute set to the value expression: *SI_height(content)*.

        vi) the *SI_width* attribute set to the value expression: *SI_width(content)*.

        vii) If *maintainFeatures* is equal 1 (one), then the *SI_StillImage* value regenerates the values of the all the implementation-dependent image feature attributes values.

### 5.1.3 SI_setContent Method

**Purpose**

Update the *SI_StillImage* content.

**Definition**

```
CREATE METHOD SI_setContent
   (content BINARY LARGE OBJECT(SI_MaxContentLength))
   RETURNS SI_StillImage
   FOR SI_StillImage
   BEGIN
      DECLARE NullInstance CONDITION FOR SQLSTATE '2202D';
      DECLARE InvalidInput CONDITION FOR SQLSTATE '2FF01';

      IF SELF IS NULL THEN
         SIGNAL NullInstance
            SET MESSAGE_TEXT = 'null image value';
      END IF;
      IF TRIM(BOTH ' ' FROM SI_format(content)) <>
            TRIM(BOTH ' ' FROM SELF.SI_format) THEN
         SIGNAL InvalidInput
            SET MESSAGE_TEXT = 'incorrect image format';
      END IF;
      RETURN
         CASE
            WHEN SELF.SI_retainFeatures = 1 THEN
               SELF.
                  SI_content(content).
                  SI_contentLength(LENGTH(content)).
                  SI_height(SI_height(content)).
                  SI_width(SI_width(content)).
                  SI_InitFeatures()
```

```
                ELSE
                   SELF.
                      SI_content(content).
                      SI_contentLength(LENGTH(content)).
                      SI_height(SI_height(content)).
                      SI_width(SI_width(content))
             END;
       END
```

**Definitional Rules**

1) *SI_MaxContentLength* is the implementation-defined maximum length for the binary representation of the *SI_StillImage* attribute *SI_content*.

**Description**

1) The method *SI_setContent(BINARY LARGE OBJECT)* takes the following input parameter:
   a) a BINARY LARGE OBJECT value *content*.
2) If the *SI_retainFeatures* attribute is set to 1 (one), then the *SI_StillImage* value regenerates the values of the all the implementation-dependent image feature attributes values. If the *SI_retainFeatures* attribute is set to 0 (zero), then no action is necessary.

### 5.1.4 SI_changeFormat Method

**Purpose**

Convert the format of an *SI_StillImage* value and adjust affected attributes.

**Definition**

```
CREATE METHOD SI_changeFormat
   (targetFormat CHARACTER VARYING(SI_MaxFormatLength))
   RETURNS SI_StillImage
   FOR SI_StillImage
   BEGIN
      DECLARE UnsupportedConversion CONDITION FOR SQLSTATE '2FF11';
      DECLARE UnsupportedReferences CONDITION FOR SQLSTATE '2FF16';
      DECLARE localContent BINARY LARGE OBJECT(SI_MaxContentLength);

      IF SELF.SI_reference IS NOT NULL THEN
         SIGNAL UnsupportedReferences
            SET MESSAGE_TEXT =
               'format conversion not supported for references';
      END IF;
      IF NOT SI_supportedConversion(SELF.SI_format, targetFormat) = 1 THEN
         SIGNAL UnsupportedConversion
            SET MESSAGE_TEXT =
               'unsupported image format conversion specified';
      END IF;
      IF TRIM(BOTH ' ' FROM targetFormat) =
            TRIM(BOTH ' ' FROM SELF.SI_format) THEN
         RETURN SELF;
      ELSE
         SET localContent =
            SI_convert(SELF.SI_content, SELF.SI_format, targetFormat);
         RETURN
            CASE
               WHEN SELF.SI_retainFeatures = 1 THEN
                  SELF.
                     SI_content(localContent).
                     SI_contentLength(LENGTH(localContent)).
                     SI_format(targetFormat).
```

```
                        SI_height(SI_height(localContent)).
                        SI_width(SI_width(localContent)).
                        SI_InitFeatures()
                  ELSE
                     SELF.
                        SI_content(localContent).
                        SI_contentLength(LENGTH(localContent)).
                        SI_format(targetFormat).
                        SI_height(SI_height(localContent)).
                        SI_width(SI_width(localContent))
               END;
         END IF;
      END
```

**Definitional Rules**

1) *SI_MaxFormatLength* is the implementation-defined maximum length for the character representation of an image format indication.

**Description**

1) The method *SI_changeFormat(CHARACTER VARYING)* takes the following input parameter:

   a) a CHARACTER VARYING value *targetFormat*.

2) Case:

   a) If the image content is referenced by *SELF.SI_reference*, then an exception condition is raised: *SQL/MM Still Image exception – format conversion not supported for references*.

   b) If the implementation does not support the conversion between the formats *SELF.SI_format* and *targetFormat*, then an exception condition is raised: *SQL/MM Still Image exception – unsupported image format conversion specified*.

   c) If *targetFormat* and *SELF.SI_format* are equivalent, SELF is returned unchanged.

   d) Otherwise,

      i) *SELF.SI_content* is converted to the format *targetFormat*, and the values of *SELF.SI_contentLength*, *SELF.SI_format*, *SELF.SI_height*, and *SELF.SI_width* are adjusted to reflect the properties of the new value of *SELF.SI_content*.

      ii) If the *SI_retainFeatures* attribute is set to 1 (one), then the *SI_StillImage* value regenerates the values of the all the implementation-dependent image feature attributes values. If the *SI_retainFeatures* attribute is set to 0 (zero), then no action is necessary.

### 5.1.5 SI_Scale Methods

**Purpose**

Scale the content of an *SI_StillImage* value and adjust affected attributes.

**Definition**

```
   CREATE METHOD SI_Scale
      (height INTEGER,
       width INTEGER)
      RETURNS SI_StillImage
      FOR SI_StillImage
      BEGIN
         DECLARE InvalidInput CONDITION FOR SQLSTATE '2FF14';
         DECLARE UnsupportedReferences CONDITION FOR SQLSTATE '2FF17';
         DECLARE localContent BINARY LARGE OBJECT(SI_MaxContentLength);
         DECLARE scaleHeight INTEGER;
         DECLARE scaleWidth INTEGER;
      IF SELF.SI_reference IS NOT NULL THEN
         SIGNAL UnsupportedReferences
            SET MESSAGE_TEXT =
               'scaling not supported for references';
      END IF;
      SET scaleHeight =
         CASE
            WHEN height IS NULL THEN
               width * CEIL(SELF.SI_height / SELF.SI_width)
            ELSE
               height
         END;
      SET scaleWidth =
         CASE
            WHEN width IS NULL THEN
               height * CEIL(SELF.SI_width / SELF.SI_height)
            ELSE
               width
         END;
      IF scaleHeight IS NULL OR
         scaleWidth IS NULL OR
         scaleHeight < 1 OR
         scaleWidth < 1 OR
         NOT SI_supportedScale(SELF.SI_format) = 1 THEN
         SIGNAL InvalidInput
            SET MESSAGE_TEXT =
               'illegal specification for scaling';
      END IF;
      SET localContent =
         SI_scaleImageData(SELF.SI_content, scaleHeight, scaleWidth);
      RETURN
         CASE
            WHEN SELF.SI_retainFeatures = 1 THEN
               SELF.
                  SI_content(localContent).
                  SI_contentLength(LENGTH(localContent)).
                  SI_height(SI_height(localContent)).
                  SI_width(SI_width(localContent)).
                  SI_InitFeatures()
            ELSE
               SELF.
                  SI_content(localContent).
                  SI_contentLength(LENGTH(localContent)).
                  SI_height(SI_height(localContent)).
                  SI_width(SI_width(localContent))
         END;
   END
CREATE METHOD SI_Scale
   (scaleFactor DOUBLE PRECISION)
   RETURNS SI_StillImage
   FOR SI_StillImage
   RETURN SELF.SI_Scale(SELF.SI_height * scaleFactor,
      SELF.SI_width * scaleFactor)
```

**Definitional Rules**

1) *SI_MaxContentLength* is the implementation-defined maximum length of the binary representation of the *SI_StillImage* attribute *SI_content*.

**Description**

1) The method *SI_Scale(INTEGER, INTEGER)* takes the following input parameters:
   a) an INTEGER value *height*,
   b) an INTEGER value *width*.
2) Case:
   a) If the image content is referenced by *SELF.SI_reference*, then an exception condition is raised: *SQL/MM Still Image exception – scaling not supported for references*.
   b) If at least one of the parameters *height* and *width* is smaller than 1 (one), or if *SELF.SI_format* indicates that scaling is not supported for images with this format, then an exception condition is raised: *SQL/MM Still Image exception – illegal specification for scaling*.
   c) Otherwise,
      i) *SELF.SI_content* is scaled to fit into the bounding rectangle defined by the input parameters *height* and *width* by maintaining the aspect ratio of the size *SELF.SI_height* by *SELF.SI_width*. The values of *SELF.SI_contentLength*, *SELF.SI_height*, and *SELF.SI_width* are adjusted to reflect the properties of the new value of *SELF.SI_content*. The interpolation technique used to scale the image is implementation-defined.
      ii) If the *SI_retainFeatures* attribute is set to 1 (one), then the *SI_StillImage* value regenerates the values of the all the implementation-dependent image feature attributes values. If the *SI_retainFeatures* attribute is set to 0 (zero), then no action is necessary.
3) The method *SI_Scale(DOUBLE PRECISION)* takes the following input parameter:
   a) a DOUBLE PRECISION value *scaleFactor*.

### 5.1.6 SI_Resize Methods

**Purpose**

Resize the content of an *SI_StillImage* value and adjust affected attributes.

**Definition**

```
CREATE METHOD SI_Resize
    (height INTEGER,
     width INTEGER)
    RETURNS SI_StillImage
    FOR SI_StillImage
    BEGIN
        DECLARE InvalidInput CONDITION FOR SQLSTATE '2FF20';
        DECLARE UnsupportedReferences CONDITION FOR SQLSTATE '2FF21';
        DECLARE localContent BINARY LARGE OBJECT(SI_MaxContentLength);
        DECLARE resizeHeight INTEGER;
        DECLARE resizeWidth INTEGER;

        IF SELF.SI_reference IS NOT NULL THEN
            SIGNAL UnsupportedReferences
                SET MESSAGE_TEXT =
                    'resizing not supported for references';
        END IF;
        SET resizeHeight =
            CASE
                WHEN height IS NULL THEN
                    width * CEIL(SELF.SI_height / SELF.SI_width)
                ELSE
                    height
            END;
        SET resizeWidth =
            CASE
            WHEN width IS NULL THEN
               height * CEIL(SELF.SI_width / SELF.SI_height)
            ELSE
               width
         END;
      IF resizeHeight IS NULL OR
         resizeWidth IS NULL OR
         resizeHeight < 1 OR
         resizeWidth < 1 OR
         NOT SI_supportedResize(SELF.SI_format) = 1 THEN
         SIGNAL InvalidInput
            SET MESSAGE_TEXT =
               'illegal specification for resizing';
      END IF;
      SET localContent =
         SI_resizeImageData(SELF.SI_content,
            resizeHeight, resizeWidth);
      RETURN
         CASE
            WHEN SELF.SI_retainFeatures = 1 THEN
               SELF.
                  SI_content(localContent).
                  SI_contentLength(LENGTH(localContent)).
                  SI_height(resizeHeight).
                  SI_width(resizeWidth)
            ELSE
               SELF.
                  SI_content(localContent).
                  SI_contentLength(LENGTH(localContent)).
                  SI_height(resizeHeight).
                  SI_width(resizeWidth).
                  SI_InitFeatures()
         END;
   END
```

**Definitional Rules**

1) *SI_MaxContentLength* is the implementation-defined maximum length of the binary representation of the *SI_StillImage* attribute *SI_content*.

**Description**

1) The method *SI_Resize(INTEGER, INTEGER)* takes the following input parameters:
   a) an INTEGER value *height*,
   b) an INTEGER value *width*.
2) Case:
   a) If the image content is referenced by SELF.*SI_reference*, then an exception condition is raised: *SQL/MM Still Image exception – resizing not supported for references*.
   b) If at least one of the parameters *height* and *width* is smaller than 1 (one), or if SELF.*SI_format* indicates that resizing is not supported for images with this format, then an exception condition is raised: *SQL/MM Still Image exception – illegal specification for resizing*.
   c) Otherwise,
      i) SELF.*SI_content* is resized to a rectangle of size *height* by *width*. The aspect ratio of the image might change during the resizing. The values of SELF.*SI_contentLength*, SELF.*SI_height*, and SELF.*SI_width* are adjusted to reflect the properties of the new value of SELF.*SI_content*.
      ii) If the *SI_retainFeatures* attribute is set to 1 (one), then the *SI_StillImage* value regenerates the values of the all the implementation-dependent image feature attributes values. If the *SI_retainFeatures* attribute is set to 0 (zero), then no action is necessary.

### 5.1.7 SI_Rotate Method

**Purpose**

Rotate the content of an *SI_StillImage* value and adjust affected attributes.

**Definition**

```
CREATE METHOD SI_Rotate
    (angle DOUBLE PRECISION)
   RETURNS SI_StillImage
   FOR SI_StillImage
   BEGIN
      DECLARE InvalidInput CONDITION FOR SQLSTATE '2FF19';
      DECLARE UnsupportedReferences CONDITION FOR SQLSTATE '2FF18';
      DECLARE localContent BINARY LARGE OBJECT(SI_MaxContentLength);

      IF SELF.SI_reference IS NOT NULL THEN
         SIGNAL UnsupportedReferences
            SET MESSAGE_TEXT =
               'rotating not supported for references';
      END IF;
      IF NOT SI_supportedRotate(SELF.SI_format) = 1 THEN
         SIGNAL InvalidInput
            SET MESSAGE_TEXT =
               'illegal specification for rotating';
      END IF;
      SET localContent = SI_rotateImageData(SELF.SI_content, angle);
      RETURN
         CASE
            WHEN SELF.SI_retainFeatures = 1 THEN
               SELF.
                  SI_content(localContent).
                  SI_contentLength(LENGTH(localContent)).
                  SI_height(SI_height(localContent)).
                  SI_width(SI_width(localContent)).
                  SI_InitFeatures()
            ELSE
               SELF.
                  SI_content(localContent).
                  SI_contentLength(LENGTH(localContent)).
                  SI_height(SI_height(localContent)).
                  SI_width(SI_width(localContent))
         END;
   END
```

**Definitional Rules**

1) *SI_MaxContentLength* is the implementation-defined maximum length for the binary representation of the *SI_StillImage* attribute *SI_content*.

**Description**

1) The method *SI_Rotate(DOUBLE PRECISION)* takes the following input parameter:
   a) an DOUBLE PRECISION value *angle*.
2) Case:
   a) If the image content is referenced by *SELF.SI_reference*, then an exception condition is raised: *SQL/MM Still Image exception – rotating not supported for references*.
   b) If *SELF.SI_format* indicates that rotating is not supported for images with this format, then an exception condition is raised: *SQL/MM Still Image exception – illegal specification for rotating*.
   c) Otherwise,

i) *SELF.SI_content* is rotated by a number of degrees defined by the parameter angle. A positive angle rotates the image counter-clockwise; a negative angle rotates the image clockwise. The values of *SELF.SI_contentLength*, *SELF.SI_height*, and *SELF.SI_width* are adjusted to reflect the properties of the new value of *SELF.SI_content*. The interpolation technique used to rotate the image is implementation-defined.

ii) If the *SI_retainFeatures* attribute is set to 1 (one), then the *SI_StillImage* value regenerates the values of the all the implementation-dependent image feature attributes values. If the *SI_retainFeatures* attribute is set to 0 (zero), then no action is necessary.

### 5.1.8 SI_Thumbnail Methods

**Purpose**

The *SI_Thumbnail* method derives a thumbnail from an *SI_StillImage* value.

**Definition**

```
CREATE METHOD SI_Thumbnail()
   RETURNS SI_StillImage
   FOR SI_StillImage
   RETURN SI_Thumbnail(SI_ThumbnailHeight, SI_ThumbnailWidth)

CREATE METHOD SI_Thumbnail
    (height INTEGER,
     width INTEGER)
   RETURNS SI_StillImage
   FOR SI_StillImage
   BEGIN
      DECLARE InvalidInput CONDITION FOR SQLSTATE '2FF12';

      IF height > SELF.SI_height OR
         width > SELF.SI_height OR
         NOT SI_supportedThumbnail(SELF.SI_format) = 1 THEN
         SIGNAL InvalidInput
            SET MESSAGE_TEXT =
               'illegal specification for thumbnail generation';
      END IF;
      RETURN SI_deriveThumbnail(SELF, height, width);
   END
```

**Definitional Rules**

1) *SI_ThumbnailHeight* is the implementation-dependent height for the resulting thumbnail.

2) *SI_ThumbnailWidth* is the implementation-dependent width for the resulting thumbnail.

**Description**

1) The method *SI_Thumbnail() takes no input parameters*.

2) The method *SI_Thumbnail()* returns an *SI_StillImage* value representing the thumbnail. The values for the attributes *SI_height* and *SI_width* for the resulting thumbnail are implementation-dependent.

3) The method *SI_Thumbnail(INTEGER, INTEGER)* takes the following input parameters:

   a) an INTEGER value *height*,

   b) an INTEGER value *width*.

4) The method *SI_Thumbnail()* returns an *SI_StillImage* value representing the thumbnail. If at least one of the parameters *height* and *width* is larger than the values *SELF.SI_height* and *SELF.SI_width*, respectively, or if *SELF.SI_format* indicates that the derivation of a thumbnail from images with this format is not supported, then an exception condition is raised: *SQL/MM Still Image exception – illegal specification for thumbnail generation*.

### 5.1.9 SI_InitFeatures method

**Purpose**

Extract image features into the appropriate feature attributes and enable on-going synchronization.

**Definition**

```
CREATE METHOD SI_InitFeatures()
   RETURNS SI_StillImage
   FOR SI_StillImage
   BEGIN
      IF SELF.SI_content IS NULL OR
         SI_supportedFeature('SI_AverageColor', SELF) = 0 THEN
         -- Set implementation-dependent attributes representing
         -- average color to NULL
      ELSE
         -- Set implementation-dependent attributes representing
         -- average color to average color returned by
         -- SI_AverageColor(SELF)
      END IF;
      IF SELF.SI_content IS NULL OR
         SI_supportedFeature('SI_ColorHistogram', SELF) = 0 THEN
         -- Set implementation-dependent attributes representing
         -- color histogram to NULL
      ELSE
         -- Set implementation-dependent attributes representing
         -- color histogram to color histogram returned by
         -- SI_ColorHistogram(SELF)
      END IF;
      IF SELF.SI_content IS NULL OR
         SI_supportedFeature('SI_PositionalColor', SELF) = 0 THEN
         -- Set implementation-dependent attributes representing
         -- positional color to NULL
      ELSE
         -- Set implementation-dependent attributes representing
         -- positional color to positional color returned by
         -- SI_PositionalColor(SELF)
      END IF;
      IF SELF.SI_content IS NULL OR
         SI_supportedFeature('SI_Texture', SELF) = 0 THEN
         -- Set implementation-dependent attributes representing
         -- texture to NULL
      ELSE
         -- Set implementation-dependent attributes representing
         -- texture to texture returned by
         -- SI_Texture(SELF)
      END IF;
      RETURN SELF.SI_retainFeatures(1);
   END
```

**Description**

1) The method *SI_InitFeatures* takes no parameters.

2) Set the *SI_retainFeatures* attribute of SELF to 1 (one), indicating that the feature attributes are to be maintained whenever the value of the *SI_StillImage* value (represented by SELF) changes.

3) Compute the values of the all the implementation-dependent image feature attributes values of SELF.

### 5.1.10 SI_ClearFeatures method

**Purpose**

Clear image features from feature attributes and disable on-going synchronization.

**Definition**

```
CREATE METHOD SI_ClearFeatures()
   RETURNS SI_StillImage
   FOR SI_StillImage
   BEGIN
      --
      -- !! See Description
      --
      RETURN SELF.SI_retainFeatures(0);
   END
```

**Description**

1) The method *SI_ClearFeatures* takes no parameters.
2) Set the *SI_retainFeatures* attribute of SELF to 0 (zero), indicating that the feature attributes are not to be maintained when the value of the *SI_StillImage* value (represented by SELF) changes.
3) Set values of the all the implementation-dependent image feature attributes values of SELF to the null value.

### 5.1.11 SI_Score Methods

**Purpose**

Returns the similarity of an image feature value to a feature value given as an argument.

**Definition**

```
CREATE METHOD SI_Score
   (averageColorFeature SI_AverageColor)
   RETURNS DOUBLE PRECISION
   FOR SI_StillImage
   RETURN averageColorFeature.SI_Score(SELF)

CREATE METHOD SI_Score
   (colorHistogramFeature SI_ColorHistogram)
   RETURNS DOUBLE PRECISION
   FOR SI_StillImage
   RETURN colorHistogramFeature.SI_Score(SELF)

CREATE METHOD SI_Score
   (positionalColorFeature SI_PositionalColor)
   RETURNS DOUBLE PRECISION
   FOR SI_StillImage
   RETURN positionalColorFeature.SI_Score(SELF)

CREATE METHOD SI_Score
   (textureFeature SI_Texture)
   RETURNS DOUBLE PRECISION
   FOR SI_StillImage
   RETURN textureFeature.SI_Score(SELF)

CREATE METHOD SI_Score
   (featureList SI_FeatureList)
   RETURNS DOUBLE PRECISION
   FOR SI_StillImage
   RETURN featureList.SI_Score(SELF)
```

**Description**

1) The method *SI_Score(SI_AverageColor)* takes the following input parameter:
   a) an *SI_AverageColor* value *averageColorFeature*.

2) The method *SI_Score(SI_AverageColor)* returns a value greater than or equal to 0 (zero). The lower the returned value, the better the average color value of SELF is characterized by the average color represented by *averageColorFeature*, which is used for scoring SELF.

   Case:

   a) If SELF or *averageColorFeature* is the null value, or if SELF.*SI_content* and the result of the function invocation *SI_getDLContent(SELF.SI_reference)* is the null value, or if the average color feature is not supported for SELF, then the null value is returned.

   b) Otherwise, the exact relationship between *averageColorFeature*, SELF, and the result of *SI_Score(SI_AverageColor)* is implementation-dependent.

3) The method *SI_Score(SI_ColorHistogram)* takes the following input parameter:

   a) an *SI_ColorHistogram* value *colorHistogramFeature*.

4) The method *SI_Score(SI_ColorHistogram)* returns a value greater than or equal to 0 (zero). The lower the returned value, the better the color histogram value of SELF is characterized by the color histogram represented by *colorHistogramFeature*, which is used for scoring SELF.

   Case:

   a) If SELF or *colorHistogramFeature* is the null value, or if SELF.*SI_content* and the result of the function invocation *SI_getDLContent(SELF.SI_reference)* is the null value, or if the color histogram feature is not supported for SELF, then the null value is returned.

   b) Otherwise, the exact relationship between *colorHistogramFeature*, SELF, and the result of *SI_Score(SI_ColorHistogram)* is implementation-dependent.

5) The method *SI_Score(SI_PositionalColor)* takes the following input parameter:

   a) an *SI_PositionalColor* value *positionalColorFeature*.

6) The method *SI_Score(SI_PositionalColor)* returns a value greater than or equal to 0 (zero). For scoring SELF, SELF is effectively divided into *n* by *m* rectangles, such that the product of *n* and *m* equals *SI_NumberSections*. For each rectangle, the most significant color is implementation-defined. The lower the returned value, the better the *n* by *m* most significant colors of SELF is characterized by the most significant colors represented by *positionalColorFeature*, which is used for scoring SELF.

   NOTE 4 The way in which SELF is divided into *SI_NumberSections* rectangles is implementation-dependent, as well as *n* and *m* itself. However, the division shall be performed in the same fashion for the *SI_Score(SI_PositionalColor)* method and the method *SI_PositionalColor(SI_StillImage)*.

   Case:

   a) If SELF or *positionalColorFeature* is the null value, or if SELF.*SI_content* and the result of the function invocation *SI_getDLContent(SELF.SI_reference)* is the null value, or if the positional color feature is not supported for SELF, then the null value is returned.

   b) Otherwise, the exact relationship between *positionalColorFeature*, SELF, and the result of *SI_Score(SI_PositionalColor)* is implementation-dependent.

7) The method *SI_Score(SI_Texture)* takes the following input parameter:

   a) an *SI_Texture* value *textureFeature*.

8) The method *SI_Score(SI_Texture)* returns a value greater than or equal to 0 (zero). The lower the returned value, the better the texture value of SELF is characterized by the texture represented by *textureFeature*, which is used for scoring SELF.

   Case:

   a) If SELF or *textureFeature* is the null value, or if SELF.*SI_content* and the result of the function invocation *SI_getDLContent(SELF.SI_reference)* is the null value, or if the texture feature is not supported for SELF, then the null value is returned.

   b) Otherwise, the exact relationship between *textureFeature*, SELF, and the result of *SI_Score(SI_Texture)* is implementation-dependent.

9) The method *SI_Score(SI_FeatureList)* takes the following input parameter:

   a) an *SI_FeatureList* value *featureList*.

10) The method *SI_Score(SI_FeatureList)* returns the null value if one of the following is true:

   a) SELF is the null value.

   b) *featureList* is the null value.

   c) The values *featureList.SI_AvgClrFtr*, *featureList.SI_ClrHstgrFtr*, *featureList.SI_PstnlClrFtr*, and *featureList.SI_TextureFtr* are all the null value.

   d) The sum of *featureList.SI_AvgClrFtrWght*, *featureList.SI_ClrHstgrFtrWght*, *featureList.SI_PstnlClrFtrWght*, and *featureList.SI_TextureFtrWght* is 0 (zero).

11) The method *SI_Score(SI_FeatureList)* returns a value greater than or equal to 0 (zero). The lower the returned value, the better SELF is characterized by *featureList*, which is used for scoring SELF. Let *N* be the number of feature attributes of SELF that are not the null value. For *i* ranging from 1 (one) to *N*, let $F_i$ be the value of that attribute, and $W_i$ the value of the corresponding weight attribute. Then the result is:

$$\frac{\sum_{i=1}^{N} F_i.SI\_Score(image) \cdot W_i}{\sum_{i=1}^{N} W_i}$$

### 5.1.12 Functions Complementing SI_StillImage Methods

**Purpose**

Return a specified *SI_StillImage* value.

**Definition**

```
CREATE FUNCTION SI_mkStillImage1
   (content BINARY LARGE OBJECT(SI_MaxContentLength))
   RETURNS SI_StillImage
   LANGUAGE SQL
   DETERMINISTIC
   CONTAINS SQL
   RETURNS NULL ON NULL INPUT
   RETURN NEW SI_StillImage(content)

CREATE FUNCTION SI_mkStillImage2
   (content BINARY LARGE OBJECT(SI_MaxContentLength),
    explicitFormat CHARACTER VARYING(SI_MaxFormatLength))
   RETURNS SI_StillImage
   LANGUAGE SQL
   DETERMINISTIC
   READS SQL DATA
   RETURNS NULL ON NULL INPUT
   RETURN NEW SI_StillImage(content, explicitFormat)

CREATE FUNCTION SI_mkStillImage3
   (reference DATALINK)
   RETURNS SI_StillImage
   LANGUAGE SQL
   DETERMINISTIC
   CONTAINS SQL
   RETURNS NULL ON NULL INPUT
   RETURN NEW SI_StillImage(reference)

CREATE FUNCTION SI_mkStillImage4
   (reference DATALINK,
    explicitFormat CHARACTER VARYING(SI_MaxFormatLength))
   RETURNS SI_StillImage
   LANGUAGE SQL
   DETERMINISTIC
```

```
      READS SQL DATA
      RETURNS NULL ON NULL INPUT
      RETURN NEW SI_StillImage(reference, explicitFormat)

   CREATE FUNCTION SI_mkStillImage5
      (content BINARY LARGE OBJECT(SI_MaxFormatLength),
       maintainFeatures INTEGER)
      RETURNS SI_StillImage
      LANGUAGE SQL
      DETERMINISTIC
      READS SQL DATA
      RETURNS NULL ON NULL INPUT
      RETURN NEW SI_StillImage(content, maintainFeatures)

   CREATE FUNCTION SI_mkStillImage6
      (content BINARY LARGE OBJECT(SI_MaxFormatLength),
       explicitFormat CHARACTER VARYING(SI_MaxFormatLength),
       maintainFeatures INTEGER)
      RETURNS SI_StillImage
      LANGUAGE SQL
      DETERMINISTIC
      READS SQL DATA
      RETURNS NULL ON NULL INPUT
      RETURN NEW SI_StillImage(content, explicitFormat, maintainFeatures)
```

**Definitional Rules**

1) *SI_MaxContentLength* is the implementation-defined maximum length for the binary representation of the *SI_StillImage* attribute *SI_content*.

2) *SI_MaxFormatLength* is the implementation-defined maximum length for the character representation of an image format indication.

**Description**

1) The function *SI_mkStillImage1(BINARY LARGE OBJECT)* takes the following input parameter:

   a) a BINARY LARGE OBJECT value *content*.

2) The function *SI_mkStillImage2(BINARY LARGE OBJECT, CHARACTER VARYING)* takes the following input parameters:

   a) a BINARY LARGE OBJECT value *content*,

   b) a CHARACTER VARYING value *explicitFormat*.

3) The function *SI_mkStillImage3(DATALINK)* takes the following input parameter:

   a) a DATALINK value *reference*.

4) The function *SI_mkStillImage4(DATALINK, CHARACTER VARYING)* takes the following input parameters:

   a) a DATALINK value *reference*,

   b) a CHARACTER VARYING value *explicitFormat*.

5) The function *SI_mkStillImage5(BINARY LARGE OBJECT, INTEGER)* takes the following input parameter:

   a) a BINARY LARGE OBJECT value *content,*

   b) an INTEGER value *maintainFeatures*.

6) The function *SI_mkStillImage6(BINARY LARGE OBJECT, CHARACTER VARYING, INTEGER)* takes the following input parameters:

   a) a BINARY LARGE OBJECT value *content*,

   b) a CHARACTER VARYING value *explicitFormat*,

   c) an INTEGER value *maintainFeatures*.

### 5.1.13 SI_chgContent Function

**Purpose**

Update the *SI_StillImage* content.

**Definition**

```
CREATE FUNCTION SI_chgContent
    (image SI_StillImage,
     content BINARY LARGE OBJECT(SI_MaxContentLength))
    RETURNS SI_StillImage
    LANGUAGE SQL
    DETERMINISTIC
    CONTAINS SQL
    RETURNS NULL ON NULL INPUT
    STATIC DISPATCH
    RETURN image.SI_setContent(content)
```

**Definitional Rules**

1) *SI_MaxContentLength* is the implementation-defined maximum length for the binary representation of the *SI_StillImage* attribute *SI_content*.

**Description**

1) The function *SI_chgContent(SI_StillImage, BINARY LARGE OBJECT)* takes the following input parameters:

   a) an *SI_StillImage* value *image*,

   b) a BINARY LARGE OBJECT value *content*.

### 5.1.14 SI_convertFormat Function

**Purpose**

Convert the format of an *SI_StillImage* value and adjust affected attributes.

**Definition**

```
CREATE FUNCTION SI_convertFormat
    (image SI_StillImage,
     targetFormat CHARACTER VARYING(SI_MaxFormatLength))
    RETURNS SI_StillImage
    LANGUAGE SQL
    DETERMINISTIC
    READS SQL DATA
    RETURNS NULL ON NULL INPUT
    STATIC DISPATCH
    RETURN image.SI_changeFormat(targetFormat)
```

**Definitional Rules**

1) *SI_MaxFormatLength* is the implementation-defined maximum length for the character representation of an image format indication.

**Description**

1) The function *SI_convertFormat(SI_StillImage, CHARACTER VARYING)* takes the following input parameters:

   a) an *SI_StillImage* value *image*,

   b) a CHARACTER VARYING value *targetFormat*.

### 5.1.15 SI_scaleImage Function

**Purpose**

Scale the content of an *SI_StillImage* value and adjust affected attributes.

**Definition**

```
CREATE FUNCTION SI_scaleImage
    (image SI_StillImage,
     height INTEGER,
     width INTEGER)
    RETURNS SI_StillImage
    LANGUAGE SQL
    DETERMINISTIC
    READS SQL DATA
    CALLED ON NULL INPUT
    STATIC DISPATCH
    RETURN image.SI_Scale(height, width)
```

**Description**

1) The function *SI_scaleImage(SI_StillImage, INTEGER, INTEGER)* takes the following input parameters:

   a) an *SI_StillImage* value *image*,

   b) an INTEGER value *height*,

   c) an INTEGER value *width*.

### 5.1.16 SI_zoomImage Function

**Purpose**

Scale the content of an *SI_StillImage* value and adjust affected attributes.

**Definition**

```
CREATE FUNCTION SI_zoomImage
    (image SI_StillImage,
    scaleFactor DOUBLE PRECISION)
    RETURNS SI_StillImage
    LANGUAGE SQL
    DETERMINISTIC
    READS SQL DATA
    RETURNS NULL ON NULL INPUT
    STATIC DISPATCH
    RETURN image.SI_Scale(scaleFactor)
```

**Description**

1) The function *SI_zoomImage(SI_StillImage, DOUBLE PRECISION)* takes the following input parameters:

a) an *SI_StillImage* value *image*,

b) a DOUBLE PRECISION value *scaleFactor*.

### 5.1.17 SI_resizeImage Function

**Purpose**

Resize the content of an *SI_StillImage* value and adjust affected attributes.

**Definition**

```
CREATE FUNCTION SI_resizeImage
    (image SI_StillImage,
     height INTEGER,
    width INTEGER)
    RETURNS SI_StillImage
    LANGUAGE SQL
    DETERMINISTIC
    READS SQL DATA
    CALLED ON NULL INPUT
    STATIC DISPATCH
    RETURN image.SI_Resize(height, width)
```

**Description**

1) The function *SI_resizeImage(SI_StillImage, INTEGER, INTEGER)* takes the following input parameters:

   a) an *SI_StillImage* value *image*,

   b) an INTEGER value *height*,

   c) an INTEGER value *width*.

### 5.1.18 SI_rotateImage Function

**Purpose**

Rotate the content of an *SI_StillImage* value and adjust affected attributes.

**Definition**

```
CREATE FUNCTION SI_rotateImage
   (image SI_StillImage,
   angle DOUBLE PRECISION)
   RETURNS SI_StillImage
   LANGUAGE SQL
   DETERMINISTIC
   READS SQL DATA
   RETURNS NULL ON NULL INPUT
   STATIC DISPATCH
   RETURN image.SI_Rotate(angle)
```

**Description**

1) The function *SI_rotateImage(SI_StillImage, DOUBLE PRECISION)* takes the following input parameters:

   a) an *SI_StillImage* value *image*,

   b) a DOUBLE PRECISION value *angle*.

### 5.1.19 SI_getThmbnl Function

**Purpose**

The *SI_getThmbnl* derives a thumbnail from an *SI_StillImage* value.

**Definition**

```
CREATE FUNCTION SI_getThmbnl
   (image SI_StillImage)
   RETURNS SI_StillImage
   LANGUAGE SQL
   DETERMINISTIC
   READS SQL DATA
   CALLED ON NULL INPUT
   STATIC DISPATCH
   RETURN image.SI_Thumbnail()
```

**Description**

1) The function *SI_getThmbnl(SI_StillImage)* takes the following input parameter:

   a) an *SI_StillImage* value *image*.

### 5.1.20 SI_getSizedThmbnl Function

**Purpose**

The *SI_getSizedThmbnl* derives a thumbnail with user-supplied *height* and *width* values from an *SI_StillImage* value.

**Definition**

```
CREATE FUNCTION SI_getSizedThmbnl
    (image SI_StillImage,
     height INTEGER,
     width INTEGER)
    RETURNS SI_StillImage
    LANGUAGE SQL
    DETERMINISTIC
    READS SQL DATA
    RETURNS NULL ON NULL INPUT
    STATIC DISPATCH
    RETURN image.SI_Thumbnail(height, width)
```

**Description**

1) The function *SI_getSizedThmbnl(SI_StillImage, INTEGER, INTEGER)* takes the following input parameters:
   a) an *SI_StillImage* value *image*,
   b) an INTEGER value *height*,
   c) an INTEGER value *width*.

### 5.1.21 SI_setImageFtrs Function

**Purpose**

Extract image features into the appropriate feature attributes and enable on-going synchronization.

**Definition**

```
CREATE FUNCTION SI_setImageFtrs
    (image SI_StillImage )
    RETURNS SI_StillImage
    LANGUAGE SQL
    DETERMINISTIC
    READS SQL DATA
    RETURNS NULL ON NULL INPUT
    RETURN image.SI_InitFeatures()
```

**Description**

1) The function SI_setImageFtrs(SI_StillImage) takes the following input parameter:
   b) an *SI_StillImage* value *image*.

### 5.1.22 SI_resetImageFtrs Function

**Purpose**

Clear image features from feature attributes and disable on-going synchronization.

**Definition**

```
CREATE FUNCTION SI_resetImageFtrs
    (image SI_StillImage)
    RETURNS SI_StillImage
    LANGUAGE SQL
    DETERMINISTIC
    READS SQL DATA
    RETURNS NULL ON NULL INPUT
    RETURN image.SI_ClearFeatures()
```

**Description**

1) The function *SI_resetImageFtrs(SI_StillImage)* takes the following input parameter:
   a) an *SI_StillImage* value *image*.

### 5.1.23 Functions Complementing Observer Functions of Type SI_StillImage

**Purpose**

Obtain the value of the designated *SI_StillImage* attribute.

**Definition**

```
CREATE FUNCTION SI_getContent
    (image SI_StillImage)
    RETURNS BINARY LARGE OBJECT(SI_MaxContentLength)
    DETERMINISTIC
    CONTAINS SQL
    RETURNS NULL ON NULL INPUT
    STATIC DISPATCH
    RETURN image.SI_content

CREATE FUNCTION SI_getContentLngth
    (image SI_StillImage)
    RETURNS INTEGER
    DETERMINISTIC
    CONTAINS SQL
    RETURNS NULL ON NULL INPUT
    STATIC DISPATCH
    RETURN image.SI_contentLength

CREATE FUNCTION SI_getReference
    (image SI_StillImage)
    RETURNS DATALINK
    DETERMINISTIC
    CONTAINS SQL
    RETURNS NULL ON NULL INPUT
    STATIC DISPATCH
    RETURN image.SI_reference

CREATE FUNCTION SI_getFormat
    (image SI_StillImage)
    RETURNS CHARACTER VARYING(SI_MaxFormatLength)
    DETERMINISTIC
    CONTAINS SQL
    RETURNS NULL ON NULL INPUT
    STATIC DISPATCH
    RETURN image.SI_format

CREATE FUNCTION SI_getHeight
    (image SI_StillImage)
    RETURNS INTEGER
    DETERMINISTIC
    CONTAINS SQL
    RETURNS NULL ON NULL INPUT
    STATIC DISPATCH
    RETURN image.SI_height

CREATE FUNCTION SI_getWidth
    (image SI_StillImage)
    RETURNS INTEGER
    DETERMINISTIC
    CONTAINS SQL
    RETURNS NULL ON NULL INPUT
    STATIC DISPATCH
    RETURN image.SI_width
```

**Definitional Rules**

1) *SI_MaxContentLength* is the implementation-defined maximum length for the binary representation of the *SI_StillImage* attribute *SI_content*.
2) *SI_MaxFormatLength* is the implementation-defined maximum length for the character representation of an image format indication.

**Description**

1) The function *SI_getContent(SI_StillImage)* takes the following input parameter:
   a) an *SI_StillImage* value *image*.
2) *SI_getContent(SI_StillImage)* returns the binary representation of the attribute *SI_content* of the *SI_StillImage* value *image*.
3) The function *SI_getContentLngth(SI_StillImage)* takes the following input parameter:
   a) an *SI_StillImage* value *image*.
4) *SI_getContentLngth(SI_StillImage)* returns the length of the binary representation of the *SI_StillImage* value *image*.
5) The function *SI_getReference(SI_StillImage)* takes the following input parameter:
   a) an *SI_StillImage* value *image*.
6) *SI_getReference(SI_StillImage)* returns the datalink value referencing the image content of the *SI_StillImage* value *image*.
7) The function *SI_getFormat(SI_StillImage)* takes the following input parameter:
   a) an *SI_StillImage* value *image*.
8) *SI_getFormat(SI_StillImage)* returns the character representation of the image format indication of the *SI_StillImage* value *image*.
9) The function *SI_getHeight(SI_StillImage)* takes the following input parameter:
   a) an *SI_StillImage* value *image*.
10) *SI_getHeight(SI_StillImage)* returns the height of the *SI_StillImage* value *image*.
11) The function *SI_getWidth(SI_StillImage)* takes the following input parameter:
    a) an *SI_StillImage* value *image*.
12) *SI_getWidth(SI_StillImage)* returns the width of the *SI_StillImage* value *image*.

### 5.1.24 Functions not intended for Public Use

**Purpose**

These functions are only intended to be used within the methods *SI_StillImage*, *SI_setContent*, *SI_changeFormat*, *SI_Scale*, and *SI_Thumbnail*.

**Definition**

```
CREATE FUNCTION SI_getDLContent
   (reference DATALINK)
   RETURNS BINARY LARGE OBJECT(SI_MaxContentLength)
   DETERMINISTIC
   CONTAINS SQL
   RETURNS NULL ON NULL INPUT
   BEGIN
      --
      -- !! See Description
      --
   END
```

```
CREATE FUNCTION SI_format
   (content BINARY LARGE OBJECT(SI_MaxContentLength))
   RETURNS CHARACTER VARYING(SI_MaxFormatLength)
   DETERMINISTIC
   CONTAINS SQL
   RETURNS NULL ON NULL INPUT
   BEGIN
      --
      -- !! See Description
      --
   END

CREATE FUNCTION SI_height
   (content BINARY LARGE OBJECT(SI_MaxContentLength))
   RETURNS INTEGER
   DETERMINISTIC
   CONTAINS SQL
   RETURNS NULL ON NULL INPUT
   BEGIN
      --
      -- !! See Description
      --
   END

CREATE FUNCTION SI_width
   (content BINARY LARGE OBJECT(SI_MaxContentLength))
   RETURNS INTEGER
   DETERMINISTIC
   CONTAINS SQL
   RETURNS NULL ON NULL INPUT
   BEGIN
      --
      -- !! See Description
      --
   END


CREATE FUNCTION SI_supportedFeature
   (featureName CHARACTER VARYING(SI_MaxFeatureNameLength),
    sourceImage SI_StillImage)
   RETURNS INTEGER
   LANGUAGE SQL
   DETERMINISTIC
   READS SQL DATA
   CALLED ON NULL INPUT
   STATIC DISPATCH
   RETURN
      CASE
         WHEN EXISTS(
            SELECT *
               FROM SI_INFORMTN_SCHEMA.SI_IMAGE_FORMAT_FEATURES
               WHERE TRIM(BOTH ' ' FROM SI_FEATURE_NAME) =
                     TRIM(BOTH ' ' FROM featureName) AND
                  TRIM(BOTH ' ' FROM SI_FORMAT) =
                     TRIM(BOTH ' ' FROM sourceImage.SI_format)
            ) THEN 1
         ELSE 0
      END

CREATE FUNCTION SI_convert
   (content BINARY LARGE OBJECT(SI_MaxContentLength),
    sourceFormat CHARACTER VARYING(SI_MaxFormatLength),
    targetFormat CHARACTER VARYING(SI_MaxFormatLength))
```

```
      RETURNS BINARY LARGE OBJECT(SI_MaxContentLength)
      LANGUAGE SQL
      DETERMINISTIC
      CONTAINS SQL
      CALLED ON NULL INPUT
      BEGIN
         --
         -- !! See Description
         --
      END

   CREATE FUNCTION SI_supportedConversion
       (sourceFormat CHARACTER VARYING(SI_MaxFormatLength),
        targetFormat CHARACTER VARYING(SI_MaxFormatLength))
      RETURNS INTEGER
      LANGUAGE SQL
      DETERMINISTIC
      READS SQL DATA
      CALLED ON NULL INPUT
      RETURN
         CASE
            WHEN EXISTS(
               SELECT *
                  FROM SI_INFORMTN_SCHEMA.SI_IMAGE_FORMAT_CONVERSIONS
                  WHERE TRIM(BOTH ' ' FROM SI_SOURCE_FORMAT) =
                        TRIM(BOTH ' ' FROM sourceFormat) AND
                     TRIM(BOTH ' ' FROM SI_TARGET_FORMAT) =
                        TRIM(BOTH ' ' FROM targetFormat)
            ) THEN 1
            ELSE 0
         END


   CREATE FUNCTION SI_supportedFormat
       (specifiedFormat CHARACTER VARYING(SI_MaxFormatLength))
      RETURNS INTEGER
      LANGUAGE SQL
      DETERMINISTIC
      READS SQL DATA
      CALLED ON NULL INPUT
      RETURN
         CASE
            WHEN EXISTS(
               SELECT *
                  FROM SI_INFORMTN_SCHEMA.SI_IMAGE_FORMATS
                  WHERE TRIM(BOTH ' ' FROM SI_FORMAT) =
                        TRIM(BOTH ' ' FROM specifiedFormat)
            ) THEN 1
            ELSE 0
         END

   CREATE FUNCTION SI_scaleImageData
       (imageData BINARY LARGE OBJECT(SI_MaxContentLength),
        height INTEGER,
        width INTEGER)
      RETURNS BINARY LARGE OBJECT(SI_MaxContentLength)
      DETERMINISTIC
      CONTAINS SQL
      CALLED ON NULL INPUT
      BEGIN
         --
         -- !! See Description
         --
      END
```

```
CREATE FUNCTION SI_supportedScale
   (sourceFormat CHARACTER VARYING(SI_MaxFormatLength))
   RETURNS INTEGER
   LANGUAGE SQL
   DETERMINISTIC
   READS SQL DATA
   CALLED ON NULL INPUT
   RETURN
      CASE
         WHEN EXISTS (
            SELECT *
               FROM SI_INFORMTN_SCHEMA.SI_IMAGE_FORMATS
               WHERE TRIM(BOTH ' ' FROM SI_FORMAT) =
                  TRIM(BOTH ' ' FROM sourceFormat) AND
                  SI_SCALE = 'YES' )
            THEN 1
         ELSE 0
      END

CREATE FUNCTION SI_resizeImageData
   (imageData BINARY LARGE OBJECT(SI_MaxContentLength),
    height INTEGER,
    width INTEGER)
   RETURNS BINARY LARGE OBJECT(SI_MaxContentLength)
   DETERMINISTIC
   CONTAINS SQL
   CALLED ON NULL INPUT
   BEGIN
      --
      -- !! See Description
      --
   END

CREATE FUNCTION SI_supportedResize
   (sourceFormat CHARACTER VARYING(SI_MaxFormatLength))
   RETURNS INTEGER
   LANGUAGE SQL
   DETERMINISTIC
   READS SQL DATA
   CALLED ON NULL INPUT
   RETURN
      CASE
         WHEN EXISTS (
            SELECT *
               FROM SI_INFORMTN_SCHEMA.SI_IMAGE_FORMATS
               WHERE TRIM(BOTH ' ' FROM SI_FORMAT) =
                  TRIM(BOTH ' ' FROM sourceFormat) AND
                  SI_RESIZE = 'YES' )
            THEN 1
         ELSE 0
      END
CREATE FUNCTION SI_rotateImageData
   (imageData BINARY LARGE OBJECT(SI_MaxContentLength),
    angle DOUBLE PRECISION)
   RETURNS BINARY LARGE OBJECT(SI_MaxContentLength)
   DETERMINISTIC
   CONTAINS SQL
   CALLED ON NULL INPUT
   BEGIN
      --
      -- !! See Description
      --
   END
```

```
CREATE FUNCTION SI_supportedRotate
    (sourceFormat CHARACTER VARYING(SI_MaxFormatLength))
    RETURNS INTEGER
    LANGUAGE SQL
    DETERMINISTIC
    READS SQL DATA
    CALLED ON NULL INPUT
    RETURN
       CASE
          WHEN EXISTS (
             SELECT *
                FROM SI_INFORMTN_SCHEMA.SI_IMAGE_FORMATS
                WHERE TRIM(BOTH ' ' FROM SI_FORMAT) =
                   TRIM(BOTH ' ' FROM sourceFormat) AND
                   SI_ROTATE = 'YES' )
             THEN 1
          ELSE 0
       END

CREATE FUNCTION SI_deriveThumbnail
    (image SI_StillImage,
     height INTEGER,
     width INTEGER)
    RETURNS SI_StillImage
    DETERMINISTIC
    CONTAINS SQL
    CALLED ON NULL INPUT
    STATIC DISPATCH
    BEGIN
       --
       -- !! See Description
       --
    END

CREATE FUNCTION SI_supportedThumbnail
    (sourceFormat CHARACTER VARYING(SI_MaxFormatLength))
    RETURNS INTEGER
    LANGUAGE SQL
    DETERMINISTIC
    READS SQL DATA
    CALLED ON NULL INPUT
    RETURN
       CASE
          WHEN EXISTS(
             SELECT *
                FROM SI_INFORMTN_SCHEMA.SI_THUMBNAIL_FORMATS
                WHERE TRIM(BOTH ' ' FROM SI_FORMAT) =
                   TRIM(BOTH ' ' FROM sourceFormat)
          ) THEN 1
       ELSE 0
    END
```

### Definitional Rules

1) *SI_MaxContentLength* is the implementation-defined maximum length for the binary representation of the *SI_StillImage* attribute *SI_content*.
2) *SI_MaxFeatureNameLength* is the implementation-defined maximum length for the character representation of a basic feature name.
3) *SI_MaxFormatLength* is the implementation-defined maximum length for the character representation of an image format indication.

**Description**

1) The function *SI_getDLContent(DATALINK)* takes the following input parameter:

   a) a DATALINK value *reference*.

2) The function *SI_getDLContent(DATALINK)* returns the image content referenced by the DATALINK value *reference* as BINARY LARGE OBJECT. The result is the null value if the parameter does not reference an image.

3) The function *SI_format(BINARY LARGE OBJECT)* takes the following input parameter:

   a) a BINARY LARGE OBJECT value *content*.

4) The function *SI_format(BINARY LARGE OBJECT)* determines the image format of its parameter *image*. The result is the null value if *image* is not in an image format that is amongst the implementation-defined set of supported image formats. The determination of the image format includes a verification of the consistency of the raw image itself.

5) The function *SI_height(BINARY LARGE OBJECT)* takes the following input parameter:

   a) a BINARY LARGE OBJECT value *content*.

6) The function *SI_height(BINARY LARGE OBJECT)* determines and returns the height of its parameter *image*. The result is the null value if *image* is not in an image format is amongst the implementation-defined set of support image formats.

7) The function *SI_width(BINARY LARGE OBJECT)* takes the following input parameter:

   a) a BINARY LARGE OBJECT value *content*.

8) The function *SI_width(BINARY LARGE OBJECT)* determines and returns the width of its parameter *image*. The result is the null value if *image* is not in an image format that is amongst the implementation-defined set of supported image formats.

9) The function *SI_supportedFeature(CHARACTER VARYING, SI_StillImage)* takes the following input parameters:

   a) a CHARACTER VARYING value *featureName*,

   b) an SI_StillImage value sourceImage.

10) For the function *SI_supportedFeature(CHARACTER VARYING, SI_StillImage)*:

    Case:

    a) If the view *SI_IMAGE_FORMAT_FEATURES* of the information schema *SI_INFORMTN_SCHEMA* contains a row whose *SI_FORMAT* column value is equivalent to the value of *sourceImage.SI_format* and whose *SI_FEATURE_NAME* column value is equivalent to *featureName*, then the result of this function is 1 (one); i.e. the feature indicated by *featureName* is supported for *sourceImage*.

    b) Otherwise, the result is 0 (zero); i.e. the feature indicated by *featureName* is not supported for *sourceImage*.

11) The function *SI_convert(BINARY LARGE OBJECT, CHARACTER VARYING, CHARACTER VARYING)* takes the following input parameters:

    a) a BINARY LARGE OBJECT value *content*,

    b) a CHARACTER VARYING value *sourceFormat*,

    c) a CHARACTER VARYING value *targetFormat*.

12) The function *SI_convert(BINARY LARGE OBJECT, CHARACTER VARYING, CHARACTER VARYING)* converts the format of the image represented by the first parameter from the format indicated by the second parameter into the format indicated by the third parameter. The converted image is returned as a *BINARY LARGE OBJECT* value. If the format conversion fails, then an exception condition is raised: *SQL/MM Still Image exception – fatal error during image format conversion*.

13) The function *SI_supportedConversion(CHARACTER VARYING, CHARACTER VARYING)* takes the following input parameters:

    a) a CHARACTER VARYING value *sourceFormat*,

    b) a CHARACTER VARYING value *targetFormat*.

14) For the function *SI_supportedConversion(CHARACTER VARYING, CHARACTER VARYING):*

    Case:

    a) If the view *SI_IMAGE_FORMAT_CONVERSIONS* of the information schema *SI_INFORMTN_SCHEMA* contains a row whose *SI_SOURCE_FORMAT* and *SI_TARGET_FORMAT* column values are equivalent to *sourceFormat* and *targetFormat*, respectively, then the result of this function is 1 (one); i.e., an *SI_StillImage* value whose *SI_format* value is *sourceFormat* can be converted to an *SI_StillImage* value whose *SI_format* value is *targetFormat*.

    b) Otherwise, the result is 0 (zero); i.e., that conversion is not supported.

15) The function *SI_supportedFormat(CHARACTER VARYING)* takes the following input parameter:

    a) CHARACTER VARYING value *specifiedFormat*.

16) For the function *SI_supportedFormat(CHARACTER VARYING)*:

    Case:

    a) If the view *SI_IMAGE_FORMATS* of the information schema *SI_INFORMTN_SCHEMA* contains a row whose *SI_FORMAT* column value is equivalent to *specifiedFormat*, then the result of this function is 1 (one); i.e. *specifiedFormat* is a supported image format.

    b) Otherwise, the result is 0 (zero); i.e. *specifiedFormat* is not a supported image format.

17) The function *SI_scaleImageData(BINARY LARGE OBJECT, INTEGER, INTEGER)* takes the following input parameters:

    a) a BINARY LARGE OBJECT value *imageData*,

    b) an INTEGER value *height*,

    c) an INTEGER value *width*.

18) The function *SI_scaleImageData(BINARY LARGE OBJECT, INTEGER, INTEGER)* returns a scaled version of its parameter *imageData*.

    Case:

    a) If the scaling fails, then an exception condition is raised: SQL/MM Still Image exception – fatal error during image scaling.

    b) Otherwise, the parameters *height* and *width* specify an outer bounding rectangle for the result *RET*. The aspect ratio of the height and width of *imageData* shall be maintained for *RET*. Thus, the still image *ret* shall have a resulting height of *MIN(height, SI_height( imageData))* and width of *MIN(width, SI_width(imageData))*.

19) The function *SI_supportedScale(CHARACTER VARYING)* takes the following input parameter:

    a) a CHARACTER VARYING value *sourceFormat*.

20) For the function *SI_supportedScale(CHARACTER VARYING)*:

    Case:

    a) If the view *SI_IMAGE_FORMATS* of the Information Schema *SI_INFORMTN_SCHEMA* contains a row whose *SI_FORMAT* column value is equivalent to *sourceFormat* and whose *SI_SCALE* column value is equivalent to 'YES', then the result of this function is 1 (one); i.e. images whose format indication equals *sourceFormat* can be scaled.

    b) Otherwise, the result is 0 (zero); i.e. images whose format indication equals *sourceFormat* cannot be scaled.

21) The function *SI_resizeImageData(BINARY LARGE OBJECT, INTEGER, INTEGER)* takes the following input parameters:

    a) a BINARY LARGE OBJECT value *imageData*,

    b) an INTEGER value *height*,

    c) an INTEGER value *width*.

22) The function *SI_resizeImageData(BINARY LARGE OBJECT, INTEGER, INTEGER)* returns a resized version of its parameter *imageData*.

    Case:

    a) If the resizing fails, then an exception condition is raised: SQL/MM Still Image exception – fatal error during image resizing.

    b) Otherwise, the parameters height and width specify the *height* and *width* for the result *ret*. The aspect ratio of the *height* and *width* of *imageData* might change for *ret*.

23) The function *SI_supportedResize(CHARACTER VARYING)* takes the following input parameter:

    a) a CHARACTER VARYING value sourceFormat.

24) For the function *SI_supportedResize(CHARACTER VARYING)*:

    Case:

    a) If the view *SI_IMAGE_FORMATS* of the Information Schema *SI_INFORMTN_SCHEMA* contains a row whose *SI_FORMAT* column value is equivalent to *sourceFormat* and whose *SI_RESIZE* column value is equivalent to 'YES', then the result of this function is 1 (one); i. e. images whose format indication equals *sourceFormat* can be resized.

    b) Otherwise, the result is 0 (zero); i. e. images whose format indication equals *sourceFormat* cannot be resized.

25) The function *SI_rotateImageData(BINARY LARGE OBJECT, DOUBLE PRECISION)* takes the following input parameters:

    a) a BINARY LARGE OBJECT value *imageData*,

    b) a DOUBLE PRECISION value *angle*.

26) The function *SI_rotateImageData(BINARY LARGE OBJECT, DOUBLE PRECISION)* returns a rotated version of its parameter *imageData*.

    Case:

    a) If the rotating fails, then an exception condition is raised: SQL/MM Still Image exception – fatal error during image rotation.

    b) Otherwise, the parameter *angle* specifies the amount of degrees the image should be rotated. A positive angle rotates the image counter-clockwise; a negative angle rotates the image clockwise. For rotations by an angle that is not a multiple of 90 degrees, the raw image is padded and the height and width of the resulting still image are adjusted. The image is not scaled together with the rotation.

27) The function *SI_supportedRotate(CHARACTER VARYING)* takes the following input parameter:

    b) a CHARACTER VARYING value *sourceFormat*.

28) For the function *SI_supportedRotate(CHARACTER VARYING)*:

    Case:

    a) If the view *SI_IMAGE_FORMAT* of the information schema *SI_INFORMTN_SCHEMA* contains a row whose *SI_FORMAT* column value is equivalent to *sourceFormat* and whose *SI_ROTATE* column value is equivalent to 'YES', then the result of this function is 1 (one); i.e. images whose format indication equals *sourceFormat* can be rotated.

    b) Otherwise, the result is 0 (zero); i.e. images whose format indication equals *sourceFormat* cannot be rotated.

29) The function *SI_deriveThumbnail(SI_StillImage, INTEGER, INTEGER)* takes the following input parameters:

    a) an SI_StillImage value image,

    b) an INTEGER value *height*,

    c) an INTEGER value *width*.

30) The function *SI_deriveThumbnail(SI_StillImage, INTEGER, INTEGER)* derives and returns a thumbnail of its parameter *image*.

    Case:

    a) If the thumbnail cannot be derived from image, then an exception condition is raised: SQL/MM Still Image exception – fatal error during thumbnail computation.

    b) Otherwise, let *ret* be the still image, which represents the derived thumbnail, that is returned by *SI_deriveThumbnail*. The attributes *ret.SI_height* and *ret.SI_width* are set to the specified *height* and *width*, respectively. The format of the resulting thumbnail *ret.SI_format* is the same as the format of the still image *image*.

31) The function *SI_supportedThumbnail(CHARACTER VARYING)* takes the following input parameter:

    a) a CHARACTER VARYING value *sourceFormat*.

32) For the function *SI_supportedThumbnail(CHARACTER VARYING)*:

    Case:

    a) If the view *SI_THUMBNAIL_FORMATS* of the information schema *SI_INFORMTN_SCHEMA* contains a row whose *SI_FORMAT* column value is equivalent to *sourceFormat*, then the result of this function is 1 (one); i.e. a thumbnail can be derived from an image whose format indication equals *sourceFormat*.

    b) Otherwise, the result is 0 (zero); i.e. a thumbnail cannot be derived from an image whose format indication equals *sourceFormat*.

# 6 Feature Types

The types in this family provide for the manipulation of still image features.

## 6.1 SI_AverageColor Type and Routines

### 6.1.1 SI_AverageColor Type

**Purpose**

Provide the definition of the feature type *SI_AverageColor* and facilities for scoring *SI_StillImage* values using values of the *SI_AverageColor* type.

**Definition**

```
CREATE TYPE SI_AverageColor
   AS (
      SI_AverageColorSpec SI_Color
   )
   INSTANTIABLE
   NOT FINAL

   CONSTRUCTOR METHOD SI_AverageColor
      (sourceImage SI_StillImage)
      RETURNS SI_AverageColor
      SELF AS RESULT
      LANGUAGE SQL
      DETERMINISTIC
      READS SQL DATA
      CALLED ON NULL INPUT,

   CONSTRUCTOR METHOD SI_AverageColor
      (averageColor SI_Color)
      RETURNS SI_AverageColor
      SELF AS RESULT
      LANGUAGE SQL
      DETERMINISTIC
      CONTAINS SQL
      CALLED ON NULL INPUT,

   METHOD SI_Score
      (image SI_StillImage)
      RETURNS DOUBLE PRECISION
      LANGUAGE SQL
      DETERMINISTIC
      READS SQL DATA
      RETURNS NULL ON NULL INPUT
```

**Definitional Rules**

1) The attribute *SI_AverageColorSpec* is not for public use. There are no GRANT statements granting EXECUTE privilege on the observer and mutator functions for the attribute *SI_AverageColorSpec*.

**Description**

1) The *SI_AverageColor* type provides for public use:
   a) a method *SI_AverageColor(SI_StillImage)*,
   b) a method *SI_AverageColor(SI_Color)*,
   c) a method *SI_Score(SI_StillImage)*,
   d) a function *SI_findAvgClr(SI_StillImage)*,
   e) a function *SI_mkAvgClr(SI_Color)*,
   f) a function *SI_ScoreByAvgClr(SI_AverageColor, SI_StillImage)*.

2) The *SI_AverageColor* type represents color values that are interpreted as average color values using the attribute:

   a) an *SI_Color value SI_AverageColorSpec*.

### 6.1.2 SI_AverageColor Methods

**Purpose**

Return a specified *SI_AverageColor* value.

**Definition**

```
CREATE CONSTRUCTOR METHOD SI_AverageColor
    (sourceImage SI_StillImage)
    RETURNS SI_AverageColor
    FOR SI_AverageColor
    BEGIN
       DECLARE InvalidInput CONDITION FOR SQLSTATE '2FF06';

       IF sourceImage IS NULL OR
          (sourceImage.SI_content IS NULL AND
           SI_getDLContent(sourceImage.SI_reference) IS NULL) OR
          NOT SI_supportedFeature('SI_AverageColor', sourceImage) = 1 THEN
          SIGNAL InvalidInput
             SET MESSAGE_TEXT =
                'bad input image; average color feature cannot be
                   determined';
       END IF;
       --
       -- !! See Description
       --
    END

CREATE CONSTRUCTOR METHOD SI_AverageColor
    (averageColor SI_Color)
    RETURNS SI_AverageColor
    FOR SI_AverageColor
    BEGIN
       DECLARE InvalidInput CONDITION FOR SQLSTATE '2FF02';

       IF averageColor IS NULL THEN
          SIGNAL InvalidInput
             SET MESSAGE_TEXT =
                'incorrect average color feature specification';
       END IF;
       RETURN SELF.
          SI_AverageColorSpec(averageColor);
    END
```

**Description**

1) The method *SI_AverageColor(SI_StillImage)* takes the following input parameter:

   a) an *SI_StillImage* value *sourceImage*.

2) If any of the following is <u>*True*</u> for an invocation of the method *SI_AverageColor(SI_StillImage)*, then an exception condition is raised: *SQL/MM Still Image exception – bad input image; average color feature cannot be determined*.

   a) *sourceImage* is the null value.

   b) *sourceImage.SI_content* is the null value and the result of the function invocation *SI_getDLContent(sourceImage.SI_reference)* is the null value.

   c) *SI_AverageColor* feature is not supported for *sourceImage*.

3) The method *SI_AverageColor(SI_StillImage)* derives an *SI_AverageColor* value from the parameter *sourceImage*. To that end, for each component that is part of the color information for the image, the points in the reference grid are summed separately and the sum for each component is divided by the number of points in the reference grid. The values for each component, thus derived, defined the resulting color value.

   NOTE 5 If the *sourceImage.SI_retainFeatures* attribute value is set to 1 (one), then the implementation-dependent attribute that represents the average color feature value in the parameter *sourceImage* may be used to construct the *SI_AverageColor* value.

4) The method *SI_AverageColor(SI_Color)* takes the following input parameter:

   a) an *SI_Color* value *averageColor*.

5) If the parameters of the method *SI_AverageColor(SI_Color)* is the null value, then an exception condition is raised: *SQL/MM Still Image exception – incorrect average color feature specification*.

### 6.1.3 SI_Score Method

**Purpose**

Determine and return the score of an *SI_StillImage* value to a given *SI_AverageColor* value.

**Definition**

```
CREATE METHOD SI_Score
   (image SI_StillImage)
   RETURNS DOUBLE PRECISION
   FOR SI_AverageColor
   BEGIN
      --
      -- !! See Description
      --
   END
```

**Description**

1) The method *SI_Score(SI_StillImage)* takes the following input parameter:

   a) an *SI_StillImage* value *image*.

2) The method *SI_Score(SI_StillImage)* returns a value greater than or equal to 0 (zero). The lower the returned value, the better the average color of *image* is characterized by the average color represented by the *SI_AverageColor* value used for scoring *image*.

   Case:

   a) If SELF or *image* is the null value, or if *image.SI_content* and the result of the function invocation *SI_getDLContent(image.SI_reference)* is the null value, or if the average color feature is not supported for *image*, then the null value is returned.

   b) Otherwise, the exact relationship between the values of *SI_AverageColor*, *SI_StillImage* and the result of *SI_Score(SI_StillImage)* is implementation-dependent.

### 6.1.4 SI_findAvgClr Function

**Purpose**

Return the *SI_AverageColor* value from an *SI_StillImage* value.

**Definition**

```
CREATE FUNCTION SI_findAvgClr
   (sourceImage SI_StillImage)
   RETURNS SI_AverageColor
   DETERMINISTIC
   READS SQL DATA
   CALLED ON NULL INPUT
   STATIC DISPATCH
   RETURN NEW SI_AverageColor(sourceImage)
```

**Description**

1) The function *SI_findAvgClr(SI_StillImage)* takes the following input parameter:

   a) an *SI_StillImage* value *sourceImage*.

### 6.1.5 SI_mkAvgClr Function

**Purpose**

Return a specified *SI_AverageColor* value.

**Definition**

```
CREATE FUNCTION SI_mkAvgClr
    (averageColor SI_Color)
   RETURNS SI_AverageColor
   DETERMINISTIC
   CONTAINS SQL
   CALLED ON NULL INPUT
   STATIC DISPATCH
   RETURN NEW SI_AverageColor(averageColor)
```

**Description**

1) The function *SI_mkAvgClr(SI_Color)* takes the following input parameter:

   a) an *SI_Color* value *averageColor*.

### 6.1.6 SI_ScoreByAvgClr Function

**Purpose**

Determine and return the score of an *SI_StillImage* value to a given *SI_AverageColor* value.

**Definition**

```
CREATE FUNCTION SI_ScoreByAvgClr
    (feature SI_AverageColor,
     image SI_StillImage)
   RETURNS DOUBLE PRECISION
   DETERMINISTIC
   READS SQL DATA
   RETURNS NULL ON NULL INPUT
   STATIC DISPATCH
   RETURN feature.SI_Score(image)
```

**Description**

1) The function *SI_ScoreByAvgClr(SI_AverageColor, SI_StillImage)* takes the following input parameters:

   a) an *SI_AverageColor* value *feature*,

   b) an *SI_StillImage* value *image*.

## 6.2 SI_ColorHistogram Type and Routines

### 6.2.1 SI_ColorHistogram Type

**Purpose**

Provide the definition of the feature type *SI_ColorHistogram* and facilities for scoring *SI_StillImage* values using values of the *SI_ColorHistogram* type.

**Definition**

```
CREATE TYPE SI_ColorHistogram
   AS (
      SI_ColorsList SI_Color ARRAY[SI_MaxHistogramLength],
      SI_FrequenciesList
         DOUBLE PRECISION ARRAY[SI_MaxHistogramLength]
   )
   INSTANTIABLE
   NOT FINAL

   CONSTRUCTOR METHOD SI_ColorHistogram
      (sourceImage SI_StillImage)
      RETURNS SI_ColorHistogram
      SELF AS RESULT
      LANGUAGE SQL
      DETERMINISTIC
      READS SQL DATA
      CALLED ON NULL INPUT,

   CONSTRUCTOR METHOD SI_ColorHistogram
      (firstColor SI_Color,
       frequency DOUBLE PRECISION)
      RETURNS SI_ColorHistogram
      SELF AS RESULT
      LANGUAGE SQL
      DETERMINISTIC
      CONTAINS SQL
      CALLED ON NULL INPUT,

   CONSTRUCTOR METHOD SI_ColorHistogram
      (colors SI_Color ARRAY[SI_MaxHistogramLength],
       frequencies DOUBLE PRECISION ARRAY[SI_MaxHistogramLength])
      RETURNS SI_ColorHistogram
      SELF AS RESULT
      LANGUAGE SQL
      DETERMINISTIC
      CONTAINS SQL
      CALLED ON NULL INPUT,

   METHOD SI_Append
      (color SI_Color,
       frequency DOUBLE PRECISION)
      RETURNS SI_ColorHistogram
      SELF AS RESULT
      LANGUAGE SQL
      DETERMINISTIC
      CONTAINS SQL
      CALLED ON NULL INPUT,

   METHOD SI_Score
      (image SI_StillImage)
      RETURNS DOUBLE PRECISION
      LANGUAGE SQL
      DETERMINISTIC
      READS SQL DATA
      RETURNS NULL ON NULL INPUT
```

**Definitional Rules**

1) *SI_MaxHistogramLength* is the implementation-defined maximum number of color/frequency pairs that are admissible in an *SI_ColorHistogram* feature value.

2) The attributes *SI_ColorsList* and *SI_FrequenciesList* are not for public use. There are no GRANT statements granting EXECUTE privilege on the observer and mutator functions for the attributes *SI_ColorsList* and *SI_FrequenciesList*.

**Description**

1) The *SI_ColorHistogram* type provides for public use:
   a) a method *SI_ColorHistogram(SI_StillImage)*,
   b) a method *SI_ColorHistogram(SI_Color, DOUBLE PRECISION)*,
   c) a method *SI_ColorHistogram(SI_Color ARRAY, DOUBLE PRECISION ARRAY)*,
   d) a method *SI_Append(SI_Color, DOUBLE PRECISION)*,
   e) a method *SI_Score(SI_StillImage),*
   f) a function *SI_findClrHstgr(SI_StillImage)*,
   g) a function *SI_mkClrHstgr(SI_Color, DOUBLE PRECISION)*,
   h) a function *SI_arrayClrHstgr(SI_Color ARRAY, DOUBLE PRECISION ARRAY)*,
   i) a function *SI_appendClrHstgr(SI_ColorHistogram, SI_Color, DOUBLE PRECISION)*,
   j) a function *SI_ScoreByClrHstgr(SI_ColorHistogram, SI_StillImage)*.
2) The *SI_ColorHistogram* type represents a sequence of color/frequency pairs using the attributes:
   a) an *SI_Color* ARRAY value *SI_ColorsList*,
   b) a DOUBLE PRECISION ARRAY value *SI_FrequenciesList*; the DOUBLE PRECISION values of the array range from 0 (zero) to 100. The *i*-th value in this array is the frequency of the *i*-th color value in *SI_ColorsList*. The arrays *SI_ColorsList* and *SI_FrequenciesList* have the same number of elements.

### 6.2.2 SI_ColorHistogram Methods

**Purpose**

Return a specified *SI_ColorHistogram* value.

**Definition**

```
CREATE CONSTRUCTOR METHOD SI_ColorHistogram
    (sourceImage SI_StillImage)
   RETURNS SI_ColorHistogram
   FOR SI_ColorHistogram
   BEGIN
      DECLARE InvalidInput CONDITION FOR SQLSTATE '2FF08';

      IF sourceImage IS NULL OR
         (sourceImage.SI_content IS NULL AND
          SI_getDLContent(sourceImage.SI_reference) IS NULL) OR
         NOT SI_supportedFeature('SI_ColorHistogram', sourceImage) = 1 THEN
         SIGNAL InvalidInput
            SET MESSAGE_TEXT =
               'bad input image; color histogram feature cannot be
                  determined';
      END IF;
      --
      -- !! See Description
      --
   END

CREATE CONSTRUCTOR METHOD SI_ColorHistogram
    (firstColor SI_Color,
     frequency DOUBLE PRECISION)
   RETURNS SI_ColorHistogram
   FOR SI_ColorHistogram
```

```
      BEGIN
         DECLARE InvalidInput CONDITION FOR SQLSTATE '2FF03';

         IF firstColor IS NULL OR
            frequency IS NULL OR
            frequency < 0.0 OR frequency > 100.0 THEN
            SIGNAL InvalidInput
               SET MESSAGE_TEXT =
                  'incorrect color histogram feature specification';
         END IF;
         RETURN SELF.
            SI_ColorsList(ARRAY[firstColor]).
            SI_FrequenciesList(ARRAY[frequency]);
      END

   CREATE CONSTRUCTOR METHOD SI_ColorHistogram
      (colors SI_Color ARRAY[SI_MaxHistogramLength],
       frequencies DOUBLE PRECISION ARRAY[SI_MaxHistogramLength])
      RETURNS SI_ColorHistogram
      FOR SI_ColorHistogram
      BEGIN
         DECLARE InvalidInput CONDITION FOR SQLSTATE '2FF03';
         DECLARE i INTEGER;

         IF colors IS NULL OR
            frequencies IS NULL OR
            CARDINALITY(colors) <> CARDINALITY(frequencies) THEN
            SIGNAL InvalidInput
               SET MESSAGE_TEXT =
                  'incorrect color histogram feature specification';
         END IF;
         SET i = 1;
         WHILE i <= CARDINALITY(frequencies) DO
            IF frequencies[i] < 0.0 OR frequencies[i] > 100.0 THEN
               SIGNAL InvalidInput
                  SET MESSAGE_TEXT =
                     'incorrect color histogram feature specification';
            END IF;
            SET i = i + 1;
         END WHILE;
         RETURN SELF.
            SI_ColorsList(colors).
            SI_FrequenciesList(frequencies);
      END
```

**Definitional Rules**

1) *SI_MaxHistogramLength* is the implementation-defined maximum number of color/frequency pairs that are admissible in an *SI_ColorHistogram* feature value.

**Description**

1) The method *SI_ColorHistogram(SI_StillImage)* takes the following input parameter:

   a) an *SI_StillImage* value *sourceImage*.

2) If any of the following is *True* for an invocation of the method *SI_ColorHistogram(SI_StillImage)*, then an exception condition is raised: *SQL/MM Still Image exception – bad input image; color histogram feature cannot be determined*.

   a) *sourceImage* is the null value.

   b) *sourceImage.SI_content* is the null value and the result of the function invocation *SI_getDLContent(image.SI_reference)* is the null value.

   c) *SI_ColorHistogram* feature is not supported for *sourceImage*.

3) The method *SI_ColorHistogram(SI_StillImage)* derives an *SI_ColorHistogram* value from the parameter *sourceImage*. To that end, the color space is divided into an implementation-dependent number *N* of areas $A_i$, each of which is represented by some color $C_i$. For *i* ranging from 1 (one) to *N*, let $F_i$ be frequency values that are initially all 0 (zero). For each point in the reference grid of *sourceImage*, $F_i$ is increased by 1 (one) if the color of that point in the reference grid represents one of the colors of $A_i$. Let F be the value that is obtained by summing up all $F_i$ values. The final result is effectively a sequence of pairs ($C_i$ , ($F_i$ / *F)*100* ).

   NOTE 6 If the *sourceImage.SI_retainFeatures* attribute value is set to 1 (one), then the implementation-dependent attribute that represents the color histogram feature value in the parameter *sourceImage* may be used to construct the *SI_ColorHistogram* value.

4) The method *SI_ColorHistogram(SI_Color, DOUBLE PRECISION)* takes the following input parameters:
   a) an *SI_Color* value *firstColor*,
   b) a DOUBLE PRECISION value *frequency*.

5) If any of the parameters of the method *SI_ColorHistogram(SI_Color, DOUBLE PRECISION)* is the null value, or if the frequency value is less than 0 (zero) or greater than 100, then an exception condition is raised: *SQL/MM Still Image exception – incorrect color histogram feature specification*.

6) The method *SI_ColorHistogram(SI_Color ARRAY, DOUBLE PRECISION ARRAY)* takes the following input parameters:
   a) an *SI_Color* ARRAY value *colors*,
   b) a DOUBLE PRECISION ARRAY value *frequencies*.

7) If any of the parameters of the method *SI_ColorHistogram(SI_Color ARRAY, DOUBLE PRECISION ARRAY)* is the null value, or if a frequency value is less than 0 (zero) or greater than 100, or if the cardinalities of the arrays *colors* and *frequencies* are different, then an exception condition is raised: *SQL/MM Still Image exception – incorrect color histogram feature specification*.

## 6.2.3 SI_Append Method

**Purpose**

Extend an *SI_ColorHistogram* value by another (*SI_Color*, DOUBLE PRECISION) pair.

**Definition**

```
CREATE METHOD SI_Append
    (color SI_Color,
     frequency DOUBLE PRECISION)
    RETURNS SI_ColorHistogram
    FOR SI_ColorHistogram
    BEGIN
       DECLARE InvalidInput CONDITION FOR SQLSTATE '2FF03';

       IF CARDINALITY(SELF.SI_ColorsList) = SI_MaxHistogramLength OR
          color IS NULL OR
          frequency IS NULL OR
          frequency < 0.0 OR frequency > 100.0 THEN
          SIGNAL InvalidInput
             SET MESSAGE_TEXT =
                'incorrect color histogram feature specification';
       END IF;
       SET SELF.SI_ColorsList =
          SELF.SI_ColorsList || ARRAY[color];
       SET SELF.SI_FrequenciesList =
          SELF.SI_FrequenciesList || ARRAY[frequency];
       RETURN SELF;
    END
```

**Definitional Rules**

1) *SI_MaxHistogramLength* is the implementation-defined maximum number of color/frequency pairs that are admissible in an *SI_ColorHistogram* feature value.

**Description**

1) The method *SI_Append(SI_Color, DOUBLE PRECISION)* takes the following input parameters:
   a) an *SI_Color* value *color*,
   b) a DOUBLE PRECISION value *frequency*.
2) If the maximum number of color/frequency pairs is already reached, or if any of the parameters is the null value, or if the frequency value is less than 0 (zero) or greater than 100, then an exception condition is raised: *SQL/MM Still Image exception – incorrect color histogram feature specification*.

### 6.2.4 SI_Score Method

**Purpose**

Determine and return the score of an *SI_StillImage* value to a given *SI_ColorHistogram* value.

**Definition**

```
CREATE METHOD SI_Score
   (image SI_StillImage)
   RETURNS DOUBLE PRECISION
   FOR SI_ColorHistogram
   BEGIN
      --
      -- !! See Description
      --
   END
```

**Description**

1) The method *SI_Score(SI_StillImage)* takes the following input parameter:
   a) an *SI_StillImage* value *image*.
2) The method *SI_Score(SI_StillImage)* returns a value greater than or equal to 0 (zero). The lower the returned value, the better the color histogram of *image* is characterized by the color histogram represented by the *SI_ColorHistogram* value used for scoring *image*.

   Case:

   a) If SELF or *image* is the null value or if *image.SI_content* and the result of the function invocation *SI_getDLContent(image.SI_reference)* is the null value, or if the color histogram feature is not supported for *image*, then the null value is returned.
   b) Otherwise, the exact relationship between the values of *SI_ColorHistogram*, *SI_StillImage*, and the result of *SI_Score(SI_StillImage)* is implementation-dependent.

### 6.2.5 SI_findClrHstgr Function

**Purpose**

Return the *SI_ColorHistogram* value from an *SI_StillImage* value.

**Definition**

```
CREATE FUNCTION SI_findClrHstgr
   (sourceImage SI_StillImage)
   RETURNS SI_ColorHistogram
   DETERMINISTIC
   READS SQL DATA
   CALLED ON NULL INPUT
   STATIC DISPATCH
   RETURN NEW SI_ColorHistogram(sourceImage)
```

**Description**

1) The function *SI_findClrHstgr(SI_StillImage)* takes the following input parameter:

   a) an *SI_StillImage* value *sourceImage*.

### 6.2.6 SI_mkClrHstgr Function

**Purpose**

Return a specified *SI_ColorHistogram* value.

**Definition**

```
CREATE FUNCTION SI_mkClrHstgr
   (firstColor SI_Color,
    frequency DOUBLE PRECISION)
   RETURNS SI_ColorHistogram
   DETERMINISTIC
   CONTAINS SQL
   CALLED ON NULL INPUT
   STATIC DISPATCH
   RETURN NEW SI_ColorHistogram(firstColor, frequency)
```

**Description**

1) The function *SI_mkClrHstgr(SI_Color, DOUBLE PRECISION)* takes the following input parameters:

   a) an *SI_Color* value *firstColor*,

   b) a DOUBLE PRECISION value *frequency*.

### 6.2.7 SI_arrayClrHstgr Function

**Purpose**

Return a specified *SI_ColorHistogram* value.

**Definition**

```
CREATE FUNCTION SI_arrayClrHstgr
   (colors SI_Color ARRAY[SI_MaxHistogramLength],
    frequencies DOUBLE PRECISION ARRAY[SI_MaxHistogramLength])
   RETURNS SI_ColorHistogram
   DETERMINISTIC
   READS SQL DATA
   CALLED ON NULL INPUT
   STATIC DISPATCH
   RETURN NEW SI_ColorHistogram(colors, frequencies)
```

**Definitional Rules**

1) *SI_MaxHistogramLength* is the implementation-defined maximum number of color/frequency pairs that are admissible in an *SI_ColorHistogram* feature value.

**Description**

1) The function *SI_arrayClrHstgr(SI_Color ARRAY, DOUBLE PRECISION ARRAY)* takes the following input parameters:

   a) an *SI_Color* ARRAY value *colors*,

   b) a DOUBLE PRECISION ARRAY value *frequencies*.

### 6.2.8 SI_appendClrHstgr Function

**Purpose**

Extend an *SI_ColorHistogram* value by another (*SI_Color*, DOUBLE PRECISION) pair.

**Definition**

```
CREATE FUNCTION SI_appendClrHstgr
   (colorHistogram SI_ColorHistogram,
    color SI_Color,
    frequency  DOUBLE PRECISION)
   RETURNS SI_ColorHistogram
   DETERMINISTIC
   CONTAINS SQL
   CALLED ON NULL INPUT
   STATIC DISPATCH
   RETURN colorHistgram.SI_Append(color, frequency)
```

**Description**

1) The function *SI_appendClrHstgr(SI_ColorHistogram, SI_Color, DOUBLE PRECISION)* takes the following input parameters:
   a) an *SI_ColorHistogram* value *colorHistogram*,
   b) an *SI_Color* value *color*,
   c) a DOUBLE PRECISION value *frequency*.

### 6.2.9 SI_ScoreByClrHstgr Function

**Purpose**

Determine and return the score of an *SI_StillImage* value to a given *SI_ColorHistogram* value.

**Definition**

```
CREATE FUNCTION SI_ScoreByClrHstgr
   (feature SI_ColorHistogram,
    image SI_StillImage)
   RETURNS DOUBLE PRECISION
   DETERMINISTIC
   READS SQL DATA
   RETURNS NULL ON NULL INPUT
   STATIC DISPATCH
   RETURN feature.SI_Score(image)
```

**Description**

1) The function *SI_ScoreByClrHstgr(SI_ColorHistogram, SI_StillImage)* takes the following input parameters:
   a) an *SI_ColorHistogram* value *feature*,
   b) an *SI_StillImage* value *image*.

## 6.3 SI_PositionalColor Type and Routines

### 6.3.1 SI_PositionalColor Type

**Purpose**

Provide the definition of the feature type *SI_PositionalColor* and facilities for scoring *SI_StillImage* values using values of the *SI_PositionalColor* type.

**Definition**

```
CREATE TYPE SI_PositionalColor
   AS (
      SI_ColorPositions
         SI_Color ARRAY[SI_NumberSections]
   )
   INSTANTIABLE
   NOT FINAL
```

```
      CONSTRUCTOR METHOD SI_PositionalColor
         (sourceImage SI_StillImage)
         RETURNS SI_PositionalColor
         SELF AS RESULT
         LANGUAGE SQL
         DETERMINISTIC
         READS SQL DATA
         CALLED ON NULL INPUT,

      METHOD SI_Score
         (image SI_StillImage)
         RETURNS DOUBLE PRECISION
         LANGUAGE SQL
         DETERMINISTIC
         READS SQL DATA
         RETURNS NULL ON NULL INPUT
```

**Definitional Rules**

1) *SI_NumberSections* is the implementation-defined number of *SI_AverageColor* values that are represented by an *SI_PositionalColor* value.

2) The attribute *SI_ColorPositions* is not for public use. There are no GRANT statements granting EXECUTE privilege on the observer and mutator functions for the attribute *SI_ColorPositions*.

**Description**

1) The *SI_PositionalColor* type provides for public use:

   a) a method *SI_PositionalColor(SI_StillImage)*,

   b) a method *SI_Score(SI_StillImage)*,

   c) a function *SI_findPstnlClr(SI_StillImage)*,

   d) a function *SI_ScoreByPstnlClr(SI_PositionalColor, SI_StillImage)*.

2) The *SI_PositionalColor* type represents lists of *SI_Color* values using the attribute:

   a) an *SI_Color ARRAY* value *SI_ColorPositions*.

### 6.3.2 SI_PositionalColor Method

**Purpose**

Return the *SI_PositionalColor* value from an *SI_StillImage* value.

**Definition**

```
   CREATE CONSTRUCTOR METHOD SI_PositionalColor
      (sourceImage SI_StillImage)
      RETURNS SI_PositionalColor
      FOR SI_PositionalColor
      BEGIN
         DECLARE InvalidInput CONDITION FOR SQLSTATE '2FF07';

         IF sourceImage IS NULL OR
            (sourceImage.SI_content IS NULL AND
             SI_getDLContent(sourceImage.SI_reference) IS NULL) OR
            NOT
               SI_supportedFeature('SI_PositionalColor', sourceImage) = 1 THEN
            SIGNAL InvalidInput
               SET MESSAGE_TEXT =
                  'bad input image; positional color feature cannot be
                     determined';
         END IF;
         --
         -- !! See Description
         --
      END
```

**Description**

1) The method *SI_PositionalColor(SI_StillImage)* takes the following input parameter:

   a) an *SI_StillImage* value *sourceImage*.

2) If any of the following is *True* for an invocation of the method *SI_PositionalColor(SI_StillImage)*, then an exception condition is raised: *SQL/MM Still Image exception – bad input image; positional color feature cannot be determined*.

   a) *sourceImage* is the null value.

   b) *sourceImage.SI_content* is the null value and the result of the function invocation *SI_getDLContent(image.SI_reference)* is the null value.

   c) *SI_PositionalColor* feature is not supported for *sourceImage*.

3) The method *SI_PositionalColor(SI_StillImage)* derives an *SI_PositionalColor* value from the parameter *sourceImage*. To that end, *sourceImage* is effectively divided into *n* by *m* rectangles, such that the product of *n* and *m* equals *SI_NumberSections*. For each rectangle, the most significant color is implementation-defined. The array, thus computed, of color values which represent the most significant colors is the *SI_ColorPositions* value of the returned *SI_PositionalColor* value. Further details on the relationship between *sourceImage* and the resulting *SI_PositionalColor* value, such as the values *n* and *m*, are implementation-dependent.

   NOTE 7 If the *sourceImage.SI_retainFeatures* attribute value is set to 1 (one), then the implementation-dependent attribute that represents the positional color feature value in the parameter *sourceImage* may be used to construct the *SI_PositionalColor* value.

   NOTE 8 The color histograms, from which the color values representing the most significant color for each rectangle are derived, are determined as described in Subclause 6.2.2, "SI_ColorHistogram Methods" Description 3), for the method *SI_ColorHistogram(SI_StillImage)*.

### 6.3.3 SI_Score Method

**Purpose**

Determine and return the score of an SI_StillImage value to a given *SI_PositionalColor* value.

**Definition**

```
CREATE METHOD SI_Score
    (image SI_StillImage)
    RETURNS DOUBLE PRECISION
    FOR SI_PositionalColor
    BEGIN
        --
        -- !! See Description
        --
    END
```

**Description**

1) The method *SI_Score(SI_StillImage)* takes the following input parameter:

   a) an *SI_StillImage* value *image*.

2) The method *SI_Score(SI_StillImage)* returns a value greater than or equal to 0 (zero). For scoring *image*, that still image is effectively divided into *n* by *m* rectangles, such that the product of *n* and *m* equals *SI_NumberSections*. For each rectangle, the most significant color is implementation-defined. The lower the returned value, the better the *n* by *m* most significant colors of *image* are characterized by the most significant colors represented by the *SI_PositionalColor* value used for scoring *image*.

   NOTE 9 The way in which *image* is divided into *SI_NumberSections* of rectangles is implementation-dependent, as well *n* and *m* itself. However, the division shall be performed in the same fashion for the *SI_Score* method and the method *SI_PositionalColor*(*SI_StillImage)*.

Case:

a) If SELF or *image* is the null value, or if *image.SI_content* and the result of the function invocation *SI_getDLContent(image.SI_reference)* is the null value, or if the positional color feature is not supported for *image*, then the null value is returned.

b) Otherwise, the exact relationship between the values of *SI_PositionalColor*, *SI_StillImage* and the result of *SI_Score(SI_StillImage)* is implementation-dependent.

### 6.3.4 SI_findPstnlClr Function

**Purpose**

Return the *SI_PositionalColor* value from an *SI_StillImage* value.

**Definition**

```
CREATE FUNCTION SI_findPstnlClr
    (sourceImage SI_StillImage)
    RETURNS SI_PositionalColor
    DETERMINISTIC
    READS SQL DATA
    CALLED ON NULL INPUT
    STATIC DISPATCH
    RETURN NEW SI_PositionalColor(sourceImage)
```

**Description**

1) The function *SI_findPstnlClr(SI_StillImage)* takes the following input parameter:

   a) an *SI_StillImage* value *sourceImage*.

### 6.3.5 SI_ScoreByPstnlClr Function

**Purpose**

Determine and return the score of an *SI_StillImage* value to a given *SI_PositionalColor* value.

**Definition**

```
CREATE FUNCTION SI_ScoreByPstnlClr
    (feature SI_PositionalColor,
     image SI_StillImage)
    RETURNS DOUBLE PRECISION
    DETERMINISTIC
    READS SQL DATA
    RETURNS NULL ON NULL INPUT
    STATIC DISPATCH
    RETURN feature.SI_Score(image)
```

**Description**

1) The function *SI_ScoreByPstnlClr(SI_PositionalColor, SI_StillImage)* takes the following input parameters:

   a) an SI_PositionalColor value feature,

   b) an *SI_StillImage* value *image*.

## 6.4 SI_Texture Type and Routines

### 6.4.1 SI_Texture Type

**Purpose**

Provide the definition of the feature type *SI_Texture* and facilities for scoring *SI_StillImage* values using values of the *SI_Texture* type.

**Definition**

```
CREATE TYPE SI_Texture
   AS (
      SI_TextureEncoding CHARACTER VARYING(SI_MaxTextureLength)
   )
   INSTANTIABLE
   NOT FINAL

   CONSTRUCTOR METHOD SI_Texture
      (sourceImage SI_StillImage)
      RETURNS SI_Texture
      SELF AS RESULT
      LANGUAGE SQL
      DETERMINISTIC
      READS SQL DATA
      CALLED ON NULL INPUT,

   METHOD SI_Score
      (image SI_StillImage)
      RETURNS DOUBLE PRECISION
      LANGUAGE SQL
      DETERMINISTIC
      READS SQL DATA
      RETURNS NULL ON NULL INPUT
```

**Definitional Rules**

1) *SI_MaxTextureLength* is the implementation-defined number of bytes needed for the implementation-defined encoded representation of an *SI_Texture*.

2) The attribute *SI_TextureEncoding* is not for public use. There are no GRANT statements granting EXECUTE privilege on the observer and mutator functions for the attribute *SI_TextureEncoding*.

**Description**

1) The *SI_Texture* type provides for public use:

   a) a method *SI_Texture(SI_StillImage)*,

   b) a method *SI_Score(SI_StillImage)*,

   c) a function *SI_findTexture(SI_StillImage)*,

   d) a function *SI_ScoreByTexture(feature SI_Texture, image SI_StillImage)*.

2) The *SI_Texture* type provides for the representation of image textures using the attribute:

   a) a CHARACTER VARYING value *SI_TextureEncoding*, the values of which represents image texture characteristics such as coarseness, contrast, and directionality in an implementation-dependent fashion.

### 6.4.2 SI_Texture Method

**Purpose**

Return the *SI_Texture* value from an *SI_StillImage* value.

**Definition**

```
CREATE CONSTRUCTOR METHOD SI_Texture
   (sourceImage SI_StillImage)
   RETURNS SI_Texture
   FOR SI_Texture
   BEGIN
      DECLARE InvalidInput CONDITION FOR SQLSTATE '2FF09';
      IF sourceImage IS NULL OR
         (sourceImage.SI_content IS NULL AND
          SI_getDLContent(sourceImage.SI_reference) IS NULL) OR
         NOT SI_supportedFeature('SI_Texture', sourceImage) = 1 THEN
         SIGNAL InvalidInput
            SET MESSAGE_TEXT =
               'bad input image; texture feature cannot be determined';
      END IF;
      --
      -- !! See Description
      --
   END
```

**Definition**

1) The method *SI_Texture(SI_StillImage)* takes the following input parameter:

   a) an *SI_StillImage* value *sourceImage*.

2) If any of the following is *True* for an invocation of the method *SI_Texture(SI_StillImage)*, then an exception condition is raised: *SQL/MM Still Image exception – bad input image; texture feature cannot be determined*.

   a) *sourceImage* is the null value.

   b) *sourceImage.SI_content* is the null value and the result of the function invocation *SI_getDLContent(image.SI_reference)* is the null value.

   c) *SI_Texture* feature is not supported for *sourceImage*.

3) The method *SI_Texture(SI_StillImage)* derives an *SI_Texture* value from the parameter *sourceImage*. The relationship between *sourceImage* and the resulting *SI_Texture* value is implementation-dependent.

   NOTE 10 If the *sourceImage.SI_retainFeatures* attribute value is set to 1 (one), then the implementation-dependent attribute that represents the texture feature value in the parameter *sourceImage* may be used to construct the *SI_Texture* value.

### 6.4.3 SI_Score Method

**Purpose**

Determine and return the score of an *SI_StillImage* value to a given *SI_Texture* value.

**Definition**

```
CREATE METHOD SI_Score
   (image SI_StillImage)
   RETURNS DOUBLE PRECISION
   FOR SI_Texture
   BEGIN
      --
      -- !! See Description
      --
   END
```

**Description**

1) The method *SI_Score(SI_StillImage)* takes the following input parameter:

   a) an *SI_StillImage* value *image*.

2) The method *SI_Score(SI_StillImage)* returns a value greater than or equal to 0 (zero). The lower the returned value, the better the texture of *image* is characterized by the *SI_Texture* value used for scoring *image*.

   Case:

   a) If SELF or *image* is the null value, or if *image.SI_content* and the result of the function invocation *SI_getDLContent(image.SI_reference)* is the null value, or if the texture feature is not supported for *image*, then the null value is returned.

b) Otherwise, the exact relationship between the values of *SI_Texture*, *SI_StillImage* and the result of *SI_Score(SI_StillImage)* is implementation-dependent.

### 6.4.4 SI_findTexture Function

**Purpose**

Return the *SI_Texture* value from an *SI_StillImage* value.

**Definition**

```
CREATE FUNCTION SI_findTexture
    (sourceImage SI_StillImage)
   RETURNS SI_Texture
   DETERMINISTIC
   READS SQL DATA
   CALLED ON NULL INPUT
   STATIC DISPATCH
   RETURN NEW SI_Texture(sourceImage)
```

**Description**

1) The function *SI_findTexture(SI_StillImage)* takes the following input parameter:

   a) an *SI_StillImage* value *sourceImage*.

### 6.4.5 SI_ScoreByTexture Function

**Purpose**

Determine and return the score of an *SI_StillImage* value to a given *SI_Texture* value.

**Definition**

```
CREATE FUNCTION SI_ScoreByTexture
    (feature SI_Texture,
     image SI_StillImage)
   RETURNS DOUBLE PRECISION
   DETERMINISTIC
   READS SQL DATA
   RETURNS NULL ON NULL INPUT
   STATIC DISPATCH
   RETURN feature.SI_Score(image)
```

**Description**

1) The function *SI_ScoreByTexture(SI_Texture, SI_StillImage)* takes the following input parameters:

   a) an *SI_Texture* value *feature*,

   b) an *SI_StillImage* value *image*.

## 6.5 SI_FeatureList Type and Routines

### 6.5.1 SI_FeatureList Type

**Purpose**

Provide the definition of the type *SI_FeatureList* and facilities for scoring *SI_StillImage* values using values of the *SI_FeatureList* type.

**Definition**

```
CREATE TYPE SI_FeatureList
   AS (
      SI_AvgClrFtr SI_AverageColor,
      SI_AvgClrFtrWght DOUBLE PRECISION DEFAULT 0.0,
      SI_ClrHstgrFtr SI_ColorHistogram,
      SI_ClrHstgrFtrWght DOUBLE PRECISION DEFAULT 0.0,
      SI_PstnlClrFtr SI_PositionalColor,
      SI_PstnlClrFtrWght DOUBLE PRECISION DEFAULT 0.0,
      SI_TextureFtr SI_Texture,
      SI_TextureFtrWght DOUBLE PRECISION DEFAULT 0.0
   )
   INSTANTIABLE
   NOT FINAL

   CONSTRUCTOR METHOD SI_FeatureList
      (averageColorFeature SI_AverageColor,
       averageColorFeatureWeight DOUBLE PRECISION,
       colorHistogramFeature SI_ColorHistogram,
       colorHistogramFeatureWeight DOUBLE PRECISION,
       positionalColorFeature SI_PositionalColor,
       positionalColorFeatureWeight DOUBLE PRECISION,
       textureFeature SI_Texture,
       textureFeatureWeight DOUBLE PRECISION)
      RETURNS SI_FeatureList
      SELF AS RESULT
      LANGUAGE SQL
      DETERMINISTIC
      CONTAINS SQL
      CALLED ON NULL INPUT,

   METHOD SI_setFeature
      (averageColorFeature SI_AverageColor,
       averageColorFeatureWeight DOUBLE PRECISION)
      RETURNS SI_FeatureList
      SELF AS RESULT
      LANGUAGE SQL
      DETERMINISTIC
      CONTAINS SQL
      CALLED ON NULL INPUT,

   METHOD SI_setFeature
      (colorHistogramFeature SI_ColorHistogram,
       colorHistogramFeatureWeight DOUBLE PRECISION)
      RETURNS SI_FeatureList
      SELF AS RESULT
      LANGUAGE SQL
      DETERMINISTIC
      CONTAINS SQL
      CALLED ON NULL INPUT,

   METHOD SI_setFeature
      (positionalColorFeature SI_PositionalColor,
       positionalColorFeatureWeight DOUBLE PRECISION)
      RETURNS SI_FeatureList
      SELF AS RESULT
      LANGUAGE SQL
      DETERMINISTIC
      CONTAINS SQL
      CALLED ON NULL INPUT,

   METHOD SI_setFeature
      (textureFeature SI_Texture,
       textureFeatureWeight DOUBLE PRECISION)
      RETURNS SI_FeatureList
      SELF AS RESULT
      LANGUAGE SQL
      DETERMINISTIC
      CONTAINS SQL
      CALLED ON NULL INPUT,
   METHOD SI_Score
      (image SI_StillImage)
      RETURNS DOUBLE PRECISION
      LANGUAGE SQL
      DETERMINISTIC
      CONTAINS SQL
      RETURNS NULL ON NULL INPUT
```

**Definitional Rules**

1) There are no GRANT statements granting EXECUTE privilege on the mutator functions for the attributes *SI_AvgClrFtr*, *SI_AvgClrFtrWght*, *SI_ClrHstgrFtr*, *SI_ClrHstgrFtrWght*, *SI_PstnlClrFtr*, *SI_PstnlClrFtrWght*, *SI_TextureFtr*, and *SI_TextureFtrWght*.

**Description**

1) The *SI_FeatureList* type provides for public use:
   a) a method *SI_FeatureList(SI_AverageColor, DOUBLE PRECISION, SI_ColorHistogram, DOUBLE PRECISION, SI_PositionalColor, DOUBLE PRECISION, SI_Texture, DOUBLE PRECISION)*,
   b) a method *SI_setFeature(SI_AverageColor, DOUBLE PRECISION)*,
   c) a method *SI_setFeature(SI_ColorHistogram, DOUBLE PRECISION)*,
   d) a method *SI_setFeature(SI_PositionalColor, DOUBLE PRECISION)*,
   e) a method *SI_setFeature(SI_Texture, DOUBLE PRECISION)*,
   f) a method *SI_Score(SI_StillImage)*,
   g) a function *SI_mkFtrList(SI_AverageColor, DOUBLE PRECISION, SI_ColorHistogram, DOUBLE PRECISION, SI_PositionalColor, DOUBLE PRECISION, SI_Texture, DOUBLE PRECISION)*,
   h) a function *SI_setAvgClrFtr(SI_FeatureList, SI_AverageColor, DOUBLE PRECISION)*,
   i) a function *SI_setClrHstgrFtr(SI_FeatureList, SI_ColorHistogram, DOUBLE PRECISION)*,
   j) a function *SI_setPstnlClrFtr(SI_FeatureList, SI_PositionalColor, DOUBLE PRECISION)*,
   k) a function *SI_setTextureFtr(SI_FeatureList, SI_Texture, DOUBLE PRECISION)*,
   l) a function *SI_ScoreByFtrList(SI_FeatureList, SI_StillImage)*.
2) The *SI_FeatureList* type represents a list of weighted features using the attributes:
   a) an *SI_AverageColor* value *SI_AvgClrFtr*,
   b) a DOUBLE PRECISION value *SI_AvgClrFtrWght*,
   c) an *SI_ColorHistogram* value *SI_ClrHstgrFtr*,
   d) a DOUBLE PRECISION value *SI_ClrHstgrFtrWght*,
   e) an *SI_PositionalColor* value *SI_PstnlClrFtr*,
   f) a DOUBLE PRECISION value *SI_PstnlClrFtrWght*,
   g) an *SI_Texture* value *SI_TextureFtr*,
   h) a DOUBLE PRECISION value *SI_TextureFtrWght*.

### 6.5.2 SI_FeatureList Method

**Purpose**

Return a specified *SI_FeatureList* value.

**Definition**

```
CREATE CONSTRUCTOR METHOD SI_FeatureList
   (averageColorFeature SI_AverageColor,
    averageColorFeatureWeight DOUBLE PRECISION,
    colorHistogramFeature SI_ColorHistogram,
         colorHistogramFeatureWeight DOUBLE PRECISION,
         positionalColorFeature SI_PositionalColor,
         positionalColorFeatureWeight DOUBLE PRECISION,
         textureFeature SI_Texture,
         textureFeatureWeight DOUBLE PRECISION)
        RETURNS SI_FeatureList
        FOR SI_FeatureList
        BEGIN
           DECLARE InvalidInput CONDITION FOR SQLSTATE '2FF04';
           IF averageColorFeature IS NOT NULL AND
              (averageColorFeatureWeight IS NULL OR
               averageColorFeatureWeight < 0.0) OR
              colorHistogramFeature IS NOT NULL AND
              (colorHistogramFeatureWeight IS NULL OR
               colorHistogramFeatureWeight < 0.0) OR
              positionalColorFeature IS NOT NULL AND
              (positionalColorFeatureWeight IS NULL OR
               positionalColorFeatureWeight < 0.0) OR
              textureFeature IS NOT NULL AND
              (textureFeatureWeight IS NULL OR
               textureFeatureWeight < 0.0) THEN
              SIGNAL InvalidInput
                 SET MESSAGE_TEXT = 'incorrect feature list specification';
           END IF;
           RETURN SELF.
              SI_AvgClrFtr(averageColorFeature).
              SI_AvgClrFtrWght(
                 CASE
                    WHEN averageColorFeature IS NULL THEN 0.0
                    ELSE averageColorFeatureWeight
                 END).
              SI_ClrHstgrFtr(colorHistogramFeature).
              SI_ClrHstgrFtrWght(
                 CASE
                    WHEN colorHistogramFeature IS NULL THEN 0.0
                    ELSE colorHistogramFeatureWeight
                 END).
              SI_PstnlClrFtr(positionalColorFeature).
              SI_PstnlClrFtrWght(
                 CASE
                    WHEN positionalColorFeature IS NULL THEN 0.0
                    ELSE positionalColorFeatureWeight
                 END).
              SI_TextureFtr(textureFeature).
              SI_TextureFtrWght(
                 CASE
                    WHEN textureFeature IS NULL THEN 0.0
                    ELSE textureFeatureWeight
                 END);
        END
```

**Description**

1) The method *SI_FeatureList(SI_AverageColor, DOUBLE PRECISION, SI_ColorHistogram, DOUBLE PRECISION, SI_PositionalColor, DOUBLE PRECISION, SI_Texture, DOUBLE PRECISION)* takes the following input parameters:

   a) an *SI_AverageColor* value *averageColorFeature*,

   b) a DOUBLE PRECISION value *averageColorFeatureWeight*,

   c) an *SI_ColorHistogram* value *colorHistogramFeature*,

   d) a DOUBLE PRECISION value *colorHistogramFeatureWeight*,
   e) an *SI_PositionalColor* value *positionalColorFeature*,
   f) a DOUBLE PRECISION value *positionalColorFeatureWeight*,
   g) an *SI_Texture* value *textureFeature*,
   h) a DOUBLE PRECISION value *textureFeatureWeight*.

2) If any of the parameters averageColorFeature, colorHistogramFeature, positionalColorFeature, or textureFeature is not the null value and any of the corresponding parameters averageColorFeatureWeight, colorHistogramFeatureWeight, positionalColorFeatureWeight, or textureFeatureWeight is the null value or less than 0 (zero), then an exception condition is raised: *SQL/MM Still Image exception – incorrect feature list specification*.

3) If any of the parameters *averageColorFeature*, *colorHistogramFeature*, *positionalColorFeature*, or *textureFeature* is the null value, then the attribute value of *SI_AvgClrFtrWght*, *SI_ClrHstgrFtrWght*, *SI_PstnlClrFtrWght*, or *SI_TextureFtrWght*, respectively, is set to 0 (zero), disregarding the value of the respective parameter *averageColorFeatureWeight*, *colorHistogramFeatureWeight*, *positionalColorFeatureWeight*, or *textureFeatureWeight*.

### 6.5.3 SI_setFeature Methods

**Purpose**

Modify a designated feature attribute and the corresponding weight attribute of an *SI_FeatureList* value.

**Definition**

```
CREATE METHOD SI_setFeature
    (averageColorFeature SI_AverageColor,
     averageColorFeatureWeight DOUBLE PRECISION)
    RETURNS SI_FeatureList
    FOR SI_FeatureList
    BEGIN
       DECLARE InvalidInput CONDITION FOR SQLSTATE '2FF04';

       IF averageColorFeature IS NOT NULL AND
          (averageColorFeatureWeight IS NULL OR
           averageColorFeatureWeight < 0.0) THEN
          SIGNAL InvalidInput
             SET MESSAGE_TEXT = 'incorrect feature list specification';
       END IF;
       RETURN SELF.
          SI_AvgClrFtr(averageColorFeature).
          SI_AvgClrFtrWght(
             CASE
                WHEN averageColorFeature IS NULL THEN 0.0
                ELSE averageColorFeatureWeight
             END);
    END
CREATE METHOD SI_setFeature
    (colorHistogramFeature SI_ColorHistogram,
     colorHistogramFeatureWeight DOUBLE PRECISION)
    RETURNS SI_FeatureList
    FOR SI_FeatureList
    BEGIN
       DECLARE InvalidInput CONDITION FOR SQLSTATE '2FF04';

       IF colorHistogramFeature IS NOT NULL AND
          (colorHistogramFeatureWeight IS NULL OR
           colorHistogramFeatureWeight < 0.0) THEN
          SIGNAL InvalidInput
             SET MESSAGE_TEXT = 'incorrect feature list specification';
```

```
      END IF;
      RETURN SELF.
         SI_ClrHstgrFtr(colorHistogramFeature).
         SI_ClrHstgrFtrWght(
            CASE
               WHEN colorHistogramFeature IS NULL THEN 0.0
               ELSE colorHistogramFeatureWeight
            END);
   END

CREATE METHOD SI_setFeature
   (positionalColorFeature SI_PositionalColor,
    positionalColorFeatureWeight DOUBLE PRECISION)
   RETURNS SI_FeatureList
   FOR SI_FeatureList
   BEGIN
      DECLARE InvalidInput CONDITION FOR SQLSTATE '2FF04';

      IF positionalColorFeature IS NOT NULL AND
         (positionalColorFeatureWeight IS NULL OR
          positionalColorFeatureWeight < 0.0) THEN
         SIGNAL InvalidInput
            SET MESSAGE_TEXT = 'incorrect feature list specification';
      END IF;
      RETURN SELF.
         SI_ClrHstgrFtr(positionalColorFeature).
         SI_ClrHstgrFtrWght(
            CASE
               WHEN positionalColorFeature IS NULL THEN 0.0
               ELSE positionalColorFeatureWeight
            END);
   END

CREATE METHOD SI_setFeature
   (textureFeature SI_Texture,
    textureFeatureWeight DOUBLE PRECISION)
   RETURNS SI_FeatureList
   FOR SI_FeatureList
   BEGIN
      DECLARE InvalidInput CONDITION FOR SQLSTATE '2FF04';

      IF textureFeature IS NOT NULL AND
         (textureFeatureWeight IS NULL OR
          textureFeatureWeight < 0.0) THEN
         SIGNAL InvalidInput
            SET MESSAGE_TEXT = 'incorrect feature list specification';
      END IF;
      RETURN SELF.
         SI_ClrHstgrFtr(textureFeature).
         SI_ClrHstgrFtrWght(
         CASE
            WHEN textureFeature IS NULL THEN 0.0
            ELSE textureFeatureWeight
         END);
   END
```

**Description**

1) The method *SI_setFeature(SI_AverageColor, DOUBLE PRECISION)* takes the following input parameters:

   a) an *SI_AverageColor* value *averageColorFeature*,

   b) a DOUBLE PRECISION value *averageColorFeatureWeight*.

2) For the method *SI_setFeature(SI_AverageColor, DOUBLE PRECISION)*:

   Case:

   a) If the parameter *averageColorFeature* is not the null value and the parameter *averageColorFeatureWeight* is the null value or less than 0 (zero), then an exception condition is raised: *SQL/MM Still Image exception – incorrect feature list specification*.

   b) If the parameter *averageColorFeature* is the null value, then the attribute value of *SI_AvgClrFtrWght* is set to 0 (zero), disregarding the value of the parameter *averageColorFeatureWeight*.

3) The method *SI_setFeature(SI_ColorHistogram, DOUBLE PRECISION)* takes the following input parameters:

   a) an *SI_ColorHistogram* value *colorHistogramFeature*,

   b) a DOUBLE PRECISION value *colorHistogramFeatureWeight*.

4) For the method *SI_setFeature(SI_ColorHistogram, DOUBLE PRECISION)*:

   Case:

   a) If the parameter *colorHistogramFeature* is not the null value and the parameter *colorHistogramFeatureWeight* is the null value or less than 0 (zero), then an exception condition is raised: *SQL/MM Still Image exception – incorrect feature list specification*.

   b) If the parameter *colorHistogramFeature* is the null value, then the attribute value of *SI_ClrHstgrFtrWght* is set to 0 (zero), disregarding the value of the parameter *colorHistogramFeatureWeight*.

5) The method *SI_setFeature(SI_PositionalColor, DOUBLE PRECISION)* takes the following input parameters:

   a) an *SI_PositionalColor* value *positionalColorFeature*,

   b) a DOUBLE PRECISION value *positionalColorFeatureWeight*.

6) For the method *SI_setFeature(SI_PositionalColor, DOUBLE PRECISION)*:

   Case:

   a) If the parameter *positionalColorFeature* is not the null value and the parameter *positionalColorFeatureWeight* is the null value or is less than 0 (zero), then an exception condition is raised: *SQL/MM Still Image exception – incorrect feature list specification*.

   b) If the parameter *positionalColorFeature* is the null value, then the attribute value of *SI_PstnlClrFtrWght* is set to 0 (zero), disregarding the value of the parameter *positionalColorFeatureWeight*.

7) The method *SI_setFeature(SI_Texture, DOUBLE PRECISION)* takes the following input parameters:

   a) an *SI_Texture* value *textureFeature*,

   b) a DOUBLE PRECISION value *textureFeatureWeight*.

8) For the method *SI_setFeature(SI_Texture, DOUBLE PRECISION)*:

   Case:

   a) If the parameter *textureFeature* is not the null value and the parameter *textureFeatureWeight* is the null value or less than 0 (zero), then an exception condition is raised: *SQL/MM Still Image exception – incorrect feature list specification*.

   b) If the parameter *textureFeature* is the null value, then the attribute value of *SI_TextureFtrWght* is set to 0 (zero), disregarding the value of the parameter *textureFeatureWeight*.

### 6.5.4 SI_Score Method

**Purpose**

Determine and return the score of an *SI_StillImage* value to a given *SI_FeatureList* value.

**Definition**

```
CREATE METHOD SI_Score
   (image SI_StillImage)
   RETURNS DOUBLE PRECISION
   FOR SI_FeatureList
   BEGIN
      DECLARE totalWeight DOUBLE PRECISION;
      DECLARE scoreResult DOUBLE PRECISION;

      IF SELF.SI_AvgClrFtr IS NULL AND
         SELF.SI_ClrHstgrFtr IS NULL AND
         SELF.SI_PstnlClrFtr IS NULL AND
         SELF.SI_TextureFtr IS NULL THEN
         RETURN CAST (NULL AS DOUBLE PRECISION);
      END IF;
      SET totalWeight = SI_AvgClrFtrWght + SI_ClrHstgrFtrWght +
         SI_PstnlClrFtrWght + SI_TextureFtrWght;
      IF totalWeight = 0.0 THEN
         RETURN CAST (NULL AS DOUBLE PRECISION);
      END IF;
      SET scoreResult = 0.0;
      IF SELF.SI_AvgClrFtr IS NOT NULL THEN
         SET scoreResult = scoreResult +
            SELF.SI_AvgClrFtr.SI_Score(image) * SI_AvgClrFtrWght;
      END IF;
      IF SELF.SI_ClrHstgrFtr IS NOT NULL THEN
         SET scoreResult = scoreResult +
            SELF.ClrHstgrFtr.SI_Score(image) * SI_ClrHstgrFtrWght;
      END IF;
      IF SELF.SI_PstnlClrFtr IS NOT NULL THEN
         SET scoreResult = scoreResult +
            SELF.PstnlClrFtr.SI_Score(image) * SI_PstnlClrFtrWght;
      END IF;
      IF SELF.SI_TextureFtr IS NOT NULL THEN
         SET scoreResult = scoreResult +
            SELF.TextureFtr.SI_Score(image) * SI_TextureFtrWght;
      END IF;
      RETURN scoreResult/totalWeight;
   END
```

**Description**

1) The method *SI_Score(SI_StillImage)* takes the following input parameter:
   a) an *SI_StillImage* value *image*.

2) The method *SI_Score(SI_StillImage)* returns the null value if one of the following is true:
   a) SELF is the null value.
   b) *image* is the null value.
   c) The values *SELF.SI_AvgClrFtr*, *SELF.SI_ClrHstgrFtr*, *SELF.SI_PstnlClrFtr*, and *SELF.SI_TextureFtr* are all the null value.
   d) The sum of *SELF.SI_AvgClrFtrWght*, *SELF.SI_ClrHstgrFtrWght*, *SELF.SI_PstnlClrFtrWght*, and *SELF.SI_TextureFtrWght* is 0 (zero).

3) The method *SI_Score(SI_StillImage)* returns a value greater than or equal to 0 (zero). The lower the returned value, the better *image* is characterized by the *SI_FeatureList* value used for scoring *image*. Let *N* be the number of feature attributes of SELF that are not the null value. For *i* ranging from 1 (one) to *N*, let $F_i$ be the value of that attribute, and $W_i$ the value of the corresponding weight attribute. Then the result is:

$$\frac{\sum_{i=1}^{N} F_i.SI\_Score(image) \cdot W_i}{\sum_{i=1}^{N} W_i}$$

### 6.5.5 SI_mkFtrList Function

**Purpose**

Return a specified *SI_FeatureList* value.

**Definition**

```
CREATE FUNCTION SI_mkFtrList
   (averageColorFeature SI_AverageColor,
    averageColorFeatureWeight DOUBLE PRECISION,
    colorHistogramFeature SI_ColorHistogram,
    colorHistogramFeatureWeight DOUBLE PRECISION,
    positionalColorFeature SI_PositionalColor,
    positionalColorFeatureWeight DOUBLE PRECISION,
    textureFeature SI_Texture,
    textureFeatureWeight DOUBLE PRECISION)
   RETURNS SI_FeatureList
   DETERMINISTIC
   CONTAINS SQL
   CALLED ON NULL INPUT
   STATIC DISPATCH
   RETURN NEW SI_FeatureList
      (averageColorFeature, averageColorFeatureWeight,
       colorHistogramFeature, colorHistogramFeatureWeight,
       positionalColorFeature, positionalColorFeatureWeight,
       textureFeature, textureFeatureWeight)
```

**Description**

1) The function *SI_mkFtrList(SI_AverageColor, DOUBLE PRECISION, SI_ColorHistogram, DOUBLE PRECISION, SI_PositionalColor, DOUBLE PRECISION, SI_Texture, DOUBLE PRECISION)* takes the following input parameters:

   a) an *SI_AverageColor* value *averageColorFeature*,

   b) a DOUBLE PRECISION value *averageColorFeatureWeight*,

   c) an *SI_ColorHistogram* value *colorHistogramFeature*,

   d) a DOUBLE PRECISION value *colorHistogramFeatureWeight*,

   e) an *SI_PositionalColor* value *positionalColorFeature*,

   f) a DOUBLE PRECISION value *positionalColorFeatureWeight*,

   g) an *SI_Texture* value *textureFeature*,

   h) a DOUBLE PRECISION value *textureFeatureWeight*.

### 6.5.6 SI_ScoreByFtrList Function

**Purpose**

Determine and return the score of an *SI_StillImage* value to a given *SI_FeatureList* value.

**Definition**

```
CREATE FUNCTION SI_ScoreByFtrList
   (feature SI_FeatureList,
    image SI_StillImage)
   RETURNS DOUBLE PRECISION
   DETERMINISTIC
   CONTAINS SQL
   RETURNS NULL ON NULL INPUT
   STATIC DISPATCH
   RETURN feature.SI_Score(image)
```

**Description**

1) The function *SI_ScoreByFtrList(SI_FeatureList, SI_StillImage)* takes the following input parameters:

   a) an *SI_FeatureList* value *feature*,

   b) an *SI_StillImage* value *image*.

### 6.5.7 Regular Functions Complementing SI_setFeature Methods

**Purpose**

Modify a designated feature attribute and the corresponding weight attribute of an *SI_FeatureList* value.

**Definition**

```
CREATE FUNCTION SI_setAvgClrFtr
   (featureList SI_FeatureList,
    averageColorFeature SI_AverageColor,
    averageColorFeatureWeight DOUBLE PRECISION)
   RETURNS SI_FeatureList
   DETERMINISTIC
   CONTAINS SQL
   CALLED ON NULL INPUT
   STATIC DISPATCH
   RETURN featureList.
      SI_setFeature(averageColorFeature, averageColorFeatureWeight)

CREATE FUNCTION SI_setClrHstgrFtr
   (featureList SI_FeatureList,
    colorHistogramFeature SI_ColorHistogram,
    colorHistogramFeatureWeight DOUBLE PRECISION)
   RETURNS SI_FeatureList
   DETERMINISTIC
   CONTAINS SQL
   CALLED ON NULL INPUT
   STATIC DISPATCH
   RETURN featureList.
      SI_setFeature(colorHistogramFeature, colorHistogramFeatureWeight)

CREATE FUNCTION SI_setPstnlClrFtr
   (featureList SI_FeatureList,
    positionalColorFeature SI_PositionalColor,
    positionalColorFeatureWeight DOUBLE PRECISION)
   RETURNS SI_FeatureList
   DETERMINISTIC
   CONTAINS SQL
   CALLED ON NULL INPUT
   STATIC DISPATCH
   RETURN featureList.
      SI_setFeature(positionalColorFeature, positionalColorFeatureWeight)

CREATE FUNCTION SI_setTextureFtr
   (featureList SI_FeatureList,
    textureFeature SI_Texture,
    textureFeatureWeight DOUBLE PRECISION)
   RETURNS SI_FeatureList
   DETERMINISTIC
   CONTAINS SQL
   CALLED ON NULL INPUT
   STATIC DISPATCH
   RETURN featureList.
      SI_setFeature(textureFeature, textureFeatureWeight)
```

**Description**

1) The function *SI_setAvgClrFtr(SI_FeatureList, SI_AverageColor, DOUBLE PRECISION)* takes the following input parameters:
    a) an *SI_FeatureList* value *featureList*,
    b) an *SI_AverageColor* value *averageColorFeature*,
    c) a DOUBLE PRECISION value *averageColorFeatureWeight*.
2) The function *SI_setClrHstgrFtr(SI_FeatureList, SI_ColorHistogram, DOUBLE PRECISION)* takes the following input parameters:
    a) an *SI_FeatureList* value *featureList*,
    b) an *SI_ColorHistogram* value *colorHistogramFeature*,
    c) a DOUBLE PRECISION value *colorHistogramFeatureWeight*.
3) The function *SI_setPstnlClrW(SI_FeatureList, SI_PositionalColor, DOUBLE PRECISION)* takes the following input parameters:
    a) an *SI_FeatureList* value *featureList*,
    b) an *SI_PositionalColor* value *positionalColorFeature*,
    c) a DOUBLE PRECISION value *positionalColorFeatureWeight*.
4) The function *SI_setTextureW(SI_FeatureList, SI_Texture, DOUBLE PRECISION)* takes the following input parameters:
    a) an *SI_FeatureList* value *featureList*,
    b) an *SI_Texture* value *textureFeature*,
    c) a DOUBLE PRECISION value *textureFeatureWeight*.

### 6.5.8 Regular Functions Complementing Observer Functions of type SI_FeatureList

**Purpose**

Obtain the value of a designated attribute from an *SI_FeatureList* value.

**Definition**

```
CREATE FUNCTION SI_getAvgClrFtr
   (featureList SI_FeatureList)
   RETURNS SI_AverageColor
   DETERMINISTIC
   CONTAINS SQL
   RETURNS NULL ON NULL INPUT
   STATIC DISPATCH
   RETURN featureList.SI_AvgClrFtr

CREATE FUNCTION SI_getAvgClrFtrW
   (featureList SI_FeatureList)
   RETURNS DOUBLE PRECISION
   DETERMINISTIC
   CONTAINS SQL
   RETURNS NULL ON NULL INPUT
   STATIC DISPATCH
   RETURN featureList.SI_AvgClrFtrWght

CREATE FUNCTION SI_getClrHstgrFtr
   (featureList SI_FeatureList)
   RETURNS SI_ColorHistogram
   DETERMINISTIC
   CONTAINS SQL
   RETURNS NULL ON NULL INPUT
   STATIC DISPATCH
   RETURN featureList.SI_ClrHstgrFtr
```

```
CREATE FUNCTION SI_getClrHstgrFtrW
    (featureList SI_FeatureList)
   RETURNS DOUBLE PRECISION
   DETERMINISTIC
   CONTAINS SQL
   RETURNS NULL ON NULL INPUT
   STATIC DISPATCH
   RETURN featureList.SI_ClrHstgrFtrWght

CREATE FUNCTION SI_getPstnlClrFtr
    (featureList SI_FeatureList)
   RETURNS SI_PositionalColor
   DETERMINISTIC
   CONTAINS SQL
   RETURNS NULL ON NULL INPUT
   STATIC DISPATCH
   RETURN featureList.SI_PstnlClrFtr

CREATE FUNCTION SI_getPstnlClrFtrW
    (featureList SI_FeatureList)
   RETURNS DOUBLE PRECISION
   DETERMINISTIC
   CONTAINS SQL
   RETURNS NULL ON NULL INPUT
   STATIC DISPATCH
   RETURN featureList.SI_PstnlClrFtrWght


CREATE FUNCTION SI_getTextureFtr
    (featureList SI_FeatureList)
   RETURNS SI_Texture
   DETERMINISTIC
   CONTAINS SQL
   RETURNS NULL ON NULL INPUT
   STATIC DISPATCH
   RETURN featureList.SI_TextureFtr

CREATE FUNCTION SI_getTextureFtrW
    (featureList SI_FeatureList)
   RETURNS DOUBLE PRECISION
   DETERMINISTIC
   CONTAINS SQL
   RETURNS NULL ON NULL INPUT
   STATIC DISPATCH
   RETURN featureList.SI_TextureFtrWght
```

**Description**

1) The function *SI_getAvgClrFtr(SI_FeatureList)* takes the following input parameter:
   a) an *SI_FeatureList* value *featureList*.
2) The function *SI_getAvgClrFtrW(SI_FeatureList)* takes the following input parameter:
   a) an *SI_FeatureList* value *featureList*.
3) The function *SI_getClrHstgrFtr(SI_FeatureList)* takes the following input parameter:
   a) an *SI_FeatureList* value *featureList*.
4) The function *SI_getClrHstgrFtrW(SI_FeatureList)* takes the following input parameter:
   a) an *SI_FeatureList* value *featureList*.
5) The function *SI_getPstnlClrFtr(SI_FeatureList)* takes the following input parameter:
   a) an *SI_FeatureList* value *featureList*.

6) The function *SI_getPstnlClrFtrW(SI_FeatureList)* takes the following input parameter:
   a) an *SI_FeatureList* value *featureList*.
7) The function *SI_getTextureFtr(SI_FeatureList)* takes the following input parameter:
   a) an *SI_FeatureList* value *featureList*.
8) The function *SI_getTextureFtrW(SI_FeatureList)* takes the following input parameter:
   a) an *SI_FeatureList* value *featureList*.

## 6.6 Auxiliary Types and Routines

### 6.6.1 SI_Color Type

**Purpose**

Provide the definition of the type SI_Color and facilities for constructing values of this type.

**Definition**

```
CREATE TYPE SI_Color
   AS (
      --
      -- !! See Description
      --
   )
   INSTANTIABLE
   NOT FINAL

   METHOD SI_RGBColor
      (redValue INTEGER,
       greenValue INTEGER,
       blueValue INTEGER)
      RETURNS SI_Color
      SELF AS RESULT
      LANGUAGE SQL
      DETERMINISTIC
      CONTAINS SQL
      CALLED ON NULL INPUT
```

**Definitional Rules**

1) The implementation-dependent attributes are not for public use. There are no GRANT statements granting EXECUTE privilege on the observer and mutator functions for the implementation-dependent set of attributes.

**Description**

1) The *SI_Color* type provides for public use:
   a) a method *SI_RGBColor(INTEGER, INTEGER, INTEGER)*,
   b) a function *SI_mkRGBClr(INTEGER, INTEGER, INTEGER)*.

   NOTE 11 An implementation may provide additional methods or functions to construct *SI_Color* values using the representation of color values in other color spaces beside RGB.
2) The *SI_Color* type represents color values using an implementation-dependent set of attributes.
3) Values of the *SI_Color* type represent color values using an implementation-dependent color space.

### 6.6.2 SI_RGBColor Method

**Purpose**

Construct a specified *SI_Color* value using the representation of colors in the RGB color space.

**Definition**

```
CREATE METHOD SI_RGBColor
    (redValue INTEGER,
     greenValue INTEGER,
     blueValue INTEGER)
   RETURNS SI_Color
   FOR SI_Color
   BEGIN
      DECLARE InvalidInput CONDITION FOR SQLSTATE '2FF05';

      IF redValue IS NULL OR
         greenValue IS NULL OR
         blueValue IS NULL OR
         redValue < 0 OR redValue > SI_MaxRGBColor OR
         greenValue < 0 OR greenValue > SI_MaxRGBColor OR
         blueValue < 0 OR blueValue > SI_MaxRGBColor THEN
         SIGNAL InvalidInput
            SET MESSAGE_TEXT = 'incorrect color specification';
      END IF;
      --
      -- !! See Description
      --
   END
```

**Definitional Rules**

1) *SI_MaxRGBColor* is the implementation-defined maximum value for each red, green, and blue values that define the representation of a color value in the RGB color space.

**Description**

1) The method *SI_RGBColor(INTEGER, INTEGER, INTEGER)* takes the following input parameters:
   a) an INTEGER value *redValue*,
   b) an INTEGER value *greenValue*,
   c) an INTEGER value *blueValue*.
2) If any of the parameters is the null value, or if any of the input values are less than 0 (zero) or greater than *SI_MaxRGBColor*, then an exception condition is raised: *SQL/MM Still Image exception – incorrect color specification*.
3) It is implementation-dependent how the color value in the RGB color space, specified by its red, green, and blue values, is represented by the implementation-dependent set of attributes of the *SI_Color* type.

### 6.6.3 SI_mkRGBClr Function

**Purpose**

Construct a specified *SI_Color* value using the representation of colors in the RGB color space.

**Definition**

```
CREATE FUNCTION SI_mkRGBClr
    (redValue INTEGER,
     greenValue INTEGER,
     blueValue INTEGER)
   RETURNS SI_Color
   LANGUAGE SQL
   DETERMINISTIC
   CONTAINS SQL
   CALLED ON NULL INPUT
   RETURN NEW SI_RGBColor(redValue, greenValue, blueValue)
```

**Description**

1) The function *SI_mkRGBClr(INTEGER, INTEGER, INTEGER)* takes the following input parameters:
   a) an INTEGER value *redValue*,
   b) an INTEGER value *greenValue*,
   c) an INTEGER value *blueValue*.

# 7 SQL/MM Still Image Information Schema

## 7.1 Introduction

The SQL/MM Still Image Information Schema views are defined as being in a schema named *SI_INFORMTN_SCHEMA* enabling these views to be accessed in the same way as any other tables in any other schema. SELECT privilege on all of these views is granted to PUBLIC WITH GRANT OPTION so that they can be queried by any user and so that SELECT privilege can be further granted on views that reference these Information Schema views. No other privilege is granted on them so they cannot be updated.

In order to provide access to the same information that is available via the *SI_INFORMTN_SCHEMA* to an SQL-Agent in an SQL-environment where the SQL-implementation does not support Feature F391, "Long identifiers" of Part 2 of ISO/IEC 9075, alternative views are provided that use only short identifiers.

An implementation may define objects that are associated with *SI_INFORMTN_SCHEMA* that are not defined in this Clause. An implementation may also add columns to tables that are defined in this Clause.

## 7.2 SI_FEATURES view

**Purpose**

Identify the supported features.

**Definition**

```
CREATE VIEW SI_FEATURES AS
   SELECT SI_FEATURENAME, SI_FEATUREVALUE
      FROM SI_DEFINITION_SCHEMA.SI_FEATURES
```

## 7.3 SI_IMAGE_FORMATS view

**Purpose**

Identify the supported image formats and the supported functionality for images with these formats.

**Definition**

```
CREATE VIEW SI_IMAGE_FORMATS AS
   SELECT SI_FORMAT, SI_SCALE, SI_RESIZE, SI_ROTATE
      FROM SI_DEFINITION_SCHEMA.SI_IMAGE_FORMATS
```

## 7.4 SI_IMAGE_FORMAT_CONVERSIONS view

**Purpose**

Identify the source and target image formats for which an image format conversion is supported and the supported functionality for images with these formats.

**Definition**

```
CREATE VIEW SI_IMAGE_FORMAT_CONVERSIONS AS
   SELECT SI_SOURCE_FORMAT, SI_TARGET_FORMAT
      FROM SI_DEFINITION_SCHEMA.SI_IMAGE_FORMAT_CONVERSIONS
```

## 7.5 SI_IMAGE_FORMAT_FEATURES view

**Purpose**

Identify the image formats for which a certain basic feature is supported.

**Definition**

```
CREATE VIEW SI_IMAGE_FORMAT_FEATURES AS
   SELECT SI_FORMAT, SI_FEATURE_NAME
      FROM SI_DEFINITION_SCHEMA.SI_IMAGE_FORMAT_FEATURES
```

## 7.6 SI_THUMBNAIL_FORMATS view

**Purpose**

Identify the image formats from which thumbnails can be derived.

**Definition**

```
CREATE VIEW SI_THUMBNAIL_FORMATS AS
   SELECT SI_FORMAT
      FROM SI_DEFINITION_SCHEMA.SI_THUMBNAIL_FORMATS
```

## 7.7 SI_VALUES view

**Purpose**

List the implementation-defined meta-variables and their values.

**Definition**

```
CREATE VIEW SI_VALUES AS
   SELECT SI_VALUE, SI_SUPPORTED_VALUE
      FROM SI_DEFINITION_SCHEMA.SI_VALUES
```

## 7.8 Short name views

**Purpose**

Provide alternative views that use only identifiers that do not require Feature F391, "Long identifiers", of Part 2 of ISO/IEC 9075.

**Definition**

```
CREATE VIEW SI_FORMAT_CONVRSNS AS
   SELECT SI_SOURCE_FORMAT, SI_TARGET_FORMAT
      FROM SI_DEFINITION_SCHEMA.SI_IMAGE_FORMAT_CONVERSIONS

CREATE VIEW SI_IMAGE_FRMT_FTRS AS
   SELECT SI_FORMAT, SI_FEATURE_NAME
      FROM SI_DEFINITION_SCHEMA.SI_IMAGE_FORMAT_FEATURES

CREATE VIEW SI_THUMBNAIL_FRMTS AS
   SELECT SI_FORMAT
      FROM SI_DEFINITION_SCHEMA.SI_THUMBNAIL_FORMATS
```

# 8 SQL/MM Still Image Definition Schema

## 8.1 Introduction

The only purpose of the SQL/MM Still Image Definition Schema is to provide a data model to support the *SI_INFORMTN_SCHEMA* and to assist understanding.

The base tables of the SQL/MM Still Image Definition Schema are defined as being in a schema named *SI_DEFINITION_SCHEMA*. The table definitions are as complete as the definitional power of ISO/IEC 9075 allows. The table definitions are supplemented with assertions where appropriate. Each description comprises three parts:

1. The function of the definition is stated.
2. The SQL definition of the object is presented as a <table definition>.
3. An explanation of the object.

The specification provides only a model of the base tables that are required, and does not imply that an implementation shall provide the functionality in the manner described in this clause.

## 8.2 SI_FEATURES base table

**Purpose**

Identify the supported features.

**Definition**

```
CREATE TABLE SI_FEATURES
    (
    SI_FEATURENAME CHARACTER VARYING(SI_MaxFeatureNameLength),
    SI_FEATUREVALUE CHARACTER VARYING(256) NOT NULL,

    CONSTRAINT SI_FEATURES_PRIMARY_KEY PRIMARY KEY(SI_FEATURENAME),
    CONSTRAINT SI_FEATURES_CHECK CHECK
       (CASE SI_FEATURENAME
          WHEN 'SI_AverageColor' THEN
             (SI_FEATUREVALUE IN ('YES'))
          WHEN 'SI_ColorHistogram' THEN
             (SI_FEATUREVALUE IN ('YES'))
          WHEN 'SI_PositionalColor' THEN
             (SI_FEATUREVALUE IN ('YES'))
          WHEN 'SI_Texture' THEN
             (SI_FEATUREVALUE IN ('YES'))
          WHEN 'SI_Interface' THEN
             (SI_FEATUREVALUE IN ('METHOD', 'FUNCTION'))
          ELSE
             UNKNOWN
       END)
    )
INSERT INTO SI_FEATURES ( SI_FEATURENAME, SI_FEATUREVALUE ) VALUES
    ( 'SI_AverageColor', 'YES' ),
    ( 'SI_ColorHistogram', 'YES' ),
    ( 'SI_PositionalColor', 'YES' ),
    ( 'SI_Texture', 'YES' )
```

**Definitional Rules**

NOTE 12 The <else clause> in the <case expression> of the check constraint SI_FEATURES_CHECK allows implementation-defined features to be included in this table.

1) *SI_MaxFeatureNameLength* is the implementation-defined maximum length of a basic feature name.

**Description**

1) The value of *SI_FEATURENAME* is a character representation of a feature that is supported by an implementation.
2) If the value of *SI_FEATURENAME* is 'SI_AverageColor', the value of *SI_FEATUREVALUE* has the following meaning:

| | |
|---|---|
| YES | The implementation supports the *SI_AverageColor* Feature. |

3) If the value of *SI_FEATURENAME* is 'SI_ColorHistogram', the value of *SI_FEATUREVALUE* has the following meaning:

| | |
|---|---|
| YES | The implementation supports the *SI_ColorHistogram* Feature. |

4) If the value of *SI_FEATURENAME* is 'SI_PositionalColor', the value of *SI_FEATUREVALUE* has the following meaning:

| | |
|---|---|
| YES | The implementation supports the *SI_PositionalColor* Feature. |

5) If the value of *SI_FEATURENAME* is 'SI_Texture', the value of *SI_FEATUREVALUE* has the following meaning:

| | |
|---|---|
| YES | The implementation supports the *SI_Texture* Feature. |

6) If the value of *SI_FEATURENAME* is 'SI_Interface', then the values of *SI_FEATUREVALUE* have the following meanings:

METHOD The implementation supports the Method oriented approach according to item 1) in Subclause 10.1, "Requirements for conformance".

FUNCTION The implementation supports the Function oriented approach according to item 2) in Subclause 10.1, "Requirements for conformance".

## 8.3 SI_IMAGE_FORMATS base table

**Purpose**

Identify the supported image formats and the supported functionality for images with these formats.

**Definition**

```
CREATE TABLE SI_IMAGE_FORMATS
   (
   SI_FORMAT CHARACTER VARYING(SI_MaxFormatLength) NOT NULL,
   SI_SCALE CHARACTER VARYING(3),
   SI_RESIZE CHARACTER VARYING(3),
   SI_ROTATE CHARACTER VARYING(3),

   CONSTRAINT SI_FORMATS_PRIMARY_KEY PRIMARY KEY(SI_FORMAT),
   CONSTRAINT SI_FORMATS_SCALE_VALUES
      CHECK ( SI_SCALE IN ( 'YES', 'NO' ) ),
   CONSTRAINT SI_FORMATS_RESIZE_VALUES
      CHECK ( SI_RESIZE IN ( 'YES', 'NO' ) ),
   CONSTRAINT SI_FORMATS_ROTATE_VALUES
      CHECK ( SI_ROTATE IN ( 'YES', 'NO' ) )
   )
```

**Definitional Rules**

1) *SI_MaxFormatLength* is the implementation-defined maximum length for the character representation of an image format indication.

**Description**

1) The value of *SI_FORMAT* is a character representation of an image format that is supported by an implementation. For an *SI_StillImage* value whose *SI_format* value indicates a supported image format that image format is derivable from its *SI_content* value.