



ISO

International Organization for Standardization  
Organisation Internationale de Normalisation

ISO/IEC JTC 1/SC 32/WG 4

SQL/MM

**Title:** (ISO Working Draft) SQL Multimedia and Application Packages  
(SQL/MM)—Part 5: Still Image

**Project:** 1.32.04.02.05.00

**Editor:** Mark Ashworth, Email: [Mark.Ashworth@ACM.org](mailto:Mark.Ashworth@ACM.org)

**References:**

- [PER001] ISO/IEC JTC 1/SC 32/WG 4 SQL/MM:PER-001, *Minutes of Helsinki, Finland WG4 Meeting*, Paul Scarponcini with Mike Newton, October 2000,  
[ftp://jerry.ece.umassd.edu/pub/SC32/WG4/Meetings/PER\\_2001\\_04\\_Sydney\\_AUS/per001.pdf](ftp://jerry.ece.umassd.edu/pub/SC32/WG4/Meetings/PER_2001_04_Sydney_AUS/per001.pdf) or [ftp://sqlstandards.org/SC32/WG4/Meetings/PER\\_2001\\_04\\_Sydney\\_AUS/per001.pdf](ftp://sqlstandards.org/SC32/WG4/Meetings/PER_2001_04_Sydney_AUS/per001.pdf).

## 1 Editorial Notes

The attached (ISO/IEC Working Draft) SQL/MM—Part 5: Still Image document incorporates the change proposals adopted at Helsinki, Finland meeting in October 2000.

### 1.1 Conventions

The following conventions are used in the base document:

- 1) Usually the base document can be produced with change bars to indicate changes between the present document and the preceding version. New (inserted) and changed material is marked with a thin vertical bar like that next to this paragraph.
  - 2) Places in the current document where material from the preceding version has been deleted are marked with a bullet like the one next paragraph. The note beside the bullet indicates how many rules, subclauses, etc. have been deleted.
- Example of a deletion bullet.

### 1.2 Organization

In Clause 2, *Changes and Notes*, I have made a number of notes related to changes that I have made to derive this document.

In Clause 3, *Possible problems with Still Image*, there is a list of possible problems in this base document. This list should be viewed as a sort of work item list, helping identify areas that should be resolved before the base document is ready for public review. There is a further division of genuine problems versus those problems where wordsmithing is the likely answer. The second category is not viewed as sufficiently urgent to cause the base document to fail progression in a ballot for public review. Finally, there is a section for Still Image Opportunities which lists those items in which some interest has been expressed, but that are not felt to be sufficiently important to justify delaying the standard in their absence.

In Clause 4, *Still Image Outstanding Requirements and Issues*, I have listed a number of outstanding requirements and issues not yet addressed in the base document. This list should be viewed as a sort of *work item list*, helping identify requirements and issues which should be resolved before the base document is ready for public review.

In Clause 5, *Possible Problems with SQL:2002*, I have listed a number of possible problems related to SQL:1999. This list should be viewed as a work item list for ISO/IEC JTC 1/SC 32/WG 4, helping identify areas which should be resolved to assist in the development of this standard.

In Clause 6, *Guidelines for writing "user-friendly" change proposals*, I have outlined some suggested guidelines that I encourage you to use when writing change proposals. Following those guidelines will markedly help improve the quality of the document.

In Clause 7, *Machine readable change proposals*, I have outlined information on how to provide copies of change proposals on diskettes or by Internet electronic mail.

### 1.3 Electronic availability

This paper is available electronically in the following formats: PostScript (.ps) and PDF (.pdf). To access it using anonymous FTP protocols, connect to the following Internet URL:

[ftp://jerry.ece.umassd.edu/pub/SC32/WG4/Meetings/PER\\_2001\\_04\\_Sydney\\_AUS](ftp://jerry.ece.umassd.edu/pub/SC32/WG4/Meetings/PER_2001_04_Sydney_AUS)

or

[ftp://sqlstandards.org/SC32/WG4/Meetings/PER\\_2001\\_04\\_Sydney\\_AUS](ftp://sqlstandards.org/SC32/WG4/Meetings/PER_2001_04_Sydney_AUS)

Use the following file names to get the paper:

per009.pdf          *PDF*

## 2 Changes and Notes

SQL/MM—Part 5: Still Image Changes, *Helsinki, Finland, October 2000*.

[PER001] SQL/MM:PER-001, *Minutes of Helsinki, Finland WG4 Meeting*.

No changes from the minutes this round.

### Other Changes

Editorial Change:

Change true, false, and unknown to *True*, *False*, and *Unknown* to align with recent changes in ISO 9075.

### 3 Possible Problems with Still Image

Some possible problems have been observed with Still Image as defined in this document. These are noted below. Further contributions to this list are welcome. Deletions from this list (resulting from change proposals that correct the problems or from research indicating that the problems do not in fact, exist) are even more welcome. Other comments may appear in the same list.

Because of the expected dynamic nature of this list (problems being removed because they are solved, new problems being added), it will become rather confusing to have the problem numbers automatically assigned by a document processing facility. In order to reduce this confusion, I have instead assigned fixed numbers to each possible problem. These numbers will not change from printing to printing, but will instead develop gaps between numbers as problems are solved.

#### 3.1 Editor's Notes in the Base Document

In the body of the base document, I have occasionally highlighted a point that requires urgent attention thus:

<b>**Editor's Note 5-999**</b>
Text of the problem.

#### 3.2 Possible Problems

##### 3.2.1 Possible Problems in the Working Draft

##### 3.2.2 Possible Problems closed in this document

None at this time.
--------------------

##### 3.2.3 Possible Problems closed since the publication of ISO/IEC 13249-5:2000(E)

None at this time.
--------------------

##### 3.2.4 Possible Problems archived in later progression material

None at this time.
--------------------

**3.3 Wordsmithing Candidates**

**3.3.1 Wordsmithing Candidates in the Working Draft**

None at this time.

More to be supplied as required.

**3.3.2 Wordsmithing Candidates closed in this document**

None at this time.

**3.3.3 Wordsmithing Candidates closed since the publication of ISO/IEC 13249-5:2000(E)**

None at this time.

**3.3.4 Wordsmithing Candidates archived in later progression material**

None at this time.

### **3.4 Still Image Opportunities**

#### **3.4.1 Still Image Opportunities in Working Draft**

None at this time.

More to be supplied as required.

#### **3.4.2 Still Image Opportunities closed in this document**

None at this time.

More to be supplied as required.

#### **3.4.3 Still Image Opportunities closed since the publication of ISO/IEC 13249-5:2000(E)**

None at this time.

#### **3.4.4 Still Image Opportunities archived in later progression material**

None at this time.

## 4 Still Image Outstanding Requirement and Issues

Several outstanding Still Image requirements and issues have been identified. These are noted below. Further contributions to this list are welcome. Deletions from this list (resulting from change proposals which correct the issues or from research indicating that the issues do not in fact, exist) are even more welcome. Other comments may appear in the same list.

Because of the expected dynamic nature of this list (issues being removed because they are solved, new problems being added), it will become rather confusing to have the problem numbers automatically assigned by a document processing facility. In order to reduce this confusion, I have instead assigned fixed numbers to each possible problem. These numbers will not change from printing to printing, but will instead develop gaps between numbers as problems are solved.

### 4.1 SQL/MM Issues in Working Draft

None at this time.

More to be supplied as required.

### 4.2 SQL/MM Issues closed in this document

None at this time.

### 4.3 SQL/MM Issues closed since the publication of ISO/IEC 13249-5:2000(E)

None at this time.

### 4.4 SQL/MM Issues archived in later progression material

None at this time.

## 5 Possible Problems with SQL-2002

I observe some possible problems with SQL-2002. These are noted below. Further contributions to this list are welcome. Deletions from this list (resulting from ISO/IEC JTC 1/SC 32/WG 4 Change proposals which correct the problems or from research indicating that the problems do not in fact, exist) are even more welcome. Other comments may appear in the same list.

Because of the expected dynamic nature of this list (problems being removed because they are solved, new problems being added), it will become rather confusing to have the problem numbers automatically assigned by a document processing facility. In order to reduce this confusion, I have instead assigned fixed numbers to each possible problem. These numbers will not change from printing to printing, but will instead develop gaps between numbers as problems are solved.

### 5.1 SQL-2002 Issues in the Working Draft

None at this time.

More to be supplied as required.

### 5.2 SQL-2002 Issues closed in this document

None at this time.

### 5.3 SQL Issues closed since the publication of ISO/IEC 13249-5:2000(E)

None at this time.

### 5.4 SQL Issues archived in later progression material

None at this time.

## 6 Guidelines for writing "user-friendly" change proposals

**\*\*Editor's Note\*\***

Most of the following items have been adapted from the SQL Editor's notes. I wish to thank the SQL Editor for permitting me to use this information.

This chapter of the Editor's Notes offers some guidelines for writing change proposals.

**\*\*Editor's Note\*\***

I reserve the right to decline instructions to change the document when those instructions are not complete and that do not generally follow these guidelines. Specifically, if a change proposal is accepted that fails to cite the number and title of the document being changed by the proposal, the proper Clause and/or Subclause numbers and titles affected by the proposal, and the Rule numbers and partial text of Rules that are changed by the proposal, then I will not make any changes in the next edition of the base document, but will bring the paper back to the Committee for revision and completion.

Finally, I request that anyone writing a change proposal with more than two pages (not sheets!) of actual change proposal bring to the meeting where the proposal will be considered a DOS-format floppy diskette containing the plain-text of the proposal (not in any word-processor format, but in plain-text format) to give to me. Alternatively, you can email the plain-text to me before or during the meeting. Thanks!

### 6.1 Writing change proposals

Here I offer some suggestions that will make the job of writing change proposals slightly more difficult and time-consuming, because they will require that the writers pay attention to what they are writing and to some conventions that we use in the document we are changing. However, experience has thoroughly demonstrated that following these suggestions leads to easier reading of change proposals (which all of your fellow committee members must do), easier editing of those proposals into the document, and a more accurate and higher quality document.

#### 6.1.1 References

When referencing other change proposals, working papers, base documents, discussion papers, etc., always reference the ISO paper numbers whenever possible. Also, please do not reference any papers that were not distributed to ISO - alternatively, ensure that such papers are all distributed to ISO.

Always cite explicitly which base document you used as the basis for your change proposals!

#### 6.1.2 Discussion

It is helpful to the reader if the discussion part makes quite clear why the proposal is being made. It also helps if any approaches that were considered and rejected are also mentioned, together with the reasons for their rejection.

Proposal writers should strive to describe all changes and their implications in the discussion part. Readers should have to slog through the detailed language changes to understand the consequences of the change.

#### 6.1.3 Change proposals

- A) Every separate item of the proposal should be numbered.
- B) When referring to a Subclause of the document, always specify both Subclause number and the name of the Subclause (For example, Subclause 6.1, "ST\_Point Type and Routines").

- C) When inserting, deleting, or replacing text of Rules in the document, please quote enough relevant text to unambiguously identify the specific paragraph, sentence, or Rule that you are affecting. For example, if you want to replace Syntax Rule 11)c)iv)2)B) of some Subclause, it would help if you ensure that I am able to correctly identify not only B), but also 11), 11)c), 11)c)iv) and 11)c)iv)2). In some cases, several rules may start off very similarly; to aid me in making your change to the proper Rule, you might say something like "Replace Syntax Rule 11)c)iv)2)B) (the Rule that deals with the xyz in the rst) with ". . .".
- D) Where only a few words of existing text are to be changed, it is clearer both to the reviewer and the Editor to flag differences in some way, e.g., by ~~striking out deleted text~~ and underlining new text, **setting new text in boldface**. Whatever convention is used should be explained.
- E) If you are modifying existing text by deleting a few words or adding a few words, it is often least ambiguous for you to quote the new text of the entire Rule or paragraph. However, particularly when the text is lengthy, it is difficult for me to discover readily which text remains the same and which is deleted or inserted or replaced. In this case, if you can somehow "flag" the next text or place where text has been deleted, it will be helpful to me (as well as to other readers of your papers). For example, you might underline new or replaced text and put bars through deleted text. In any case, please explain the convention that you use, because not all of us will have the same conventions in our papers.
- F) In order to make the document as consistent in its typographical style as possible, please follow these suggestions:
- 1) Most "symbolic names" are in upper-case (capital) italic letters (without diacriticals or other marks) and digits. Examples: *T*, *C1*, and *SLCC*.
  - 2) In general, only SQL <key word>s should appear in upper-case roman letters.
  - 3) A very few symbolic names - those used as "indexes" or subscripts, usually - are in lower case (small) *italic* letters and digits. Examples: "the *i*-column", "*C-k*", or "at least *j* values".
  - 4) When referring to a Boolean value the values are printed as capitalized italic words that are underlined: *True*, *False*, and *Unknown*. The words "true", "false" and "unknown" are used in their normal sense.
  - 5) No parts of the text are **bolded**.
- In particular, please do not use any symbols that have a "prime" on them. For example, even in roman upper-case letters (e.g. T'), the appearance is unclear - a third-generation photocopy is quite difficult to read. If you use primes on italic upper-case letters (e.g. *T'*), the letter and the prime almost merge. Instead, try to use subscripts (e.g. *T<sub>1</sub>*) or different symbols (e.g. *T1* or *TA*).
- G) Try to use the precise BNF non-terminal that you mean in your papers.
- H) In change proposals, it's often very tempting to use "short-hand" notations (such "txn" for "transaction"). That's no problem for me in your discussions sections, of course, but it makes my editing job harder when it's in your proposal text - because I cannot just copy your proposal, but have to "translate" as I go. Please use the full, spelled-out, correct terminology in the formal proposal text.
- I) Remember the differences between Syntax and Access Rules and General Rules. The most important characteristic of this difference is that Syntax and Access Rules define the limits on *programs* that utilize standard-conforming features of SQL/MM systems (sometimes called standard-conforming programs), while General Rules specify the behavior of standard-conforming SQL/MM systems.

The difference is largely manifested by the choice of words - Syntax and Access Rules typically says things like "If X is specified, then Y shall be specified" or "Z and W shall not both be specified" (the emphasis here is the use of "shall" to specify requirements on users of the standard), while General Rules typically describe behavior of the SQL/MM system with word like "If X is not true, then an exception is raised: *some exception*" or "A table descriptor is created that specifies ..." (the emphasis here is the use of word like "is" to characterize the behavior of the system). Please don't write General Rules like "If X is specified, then Y shall be true. Otherwise, an exception ..." because that is a confusing mixture of Syntax and General Rule wording. Instead, say "If X is specified and Y is not true, then an exception ...".

- J) On the subject of exceptions and such, please recall that we have finally adopted consistent wording for the behavior with regards to the end-of-statement behavior. The two categories are: "... an exception condition is raised: *some condition*" and "... a completion condition is raised: *limited alternatives*". The only acceptable alternatives for the completion conditions are *successful completion*, *warning* (with either *no subcode* or a specific subcode), and *no data*. Furthermore, recall that the typographical convention is that the exception class and subclass phrases are separated by a dash (-) and are always both in italics.
- K) A few naming conventions are also good to remember: "data type" is two words, "database" is one word, and "datetime" is also one word. Furthermore "text in quotes" is not a BNF non-terminal but "<text in angle brackets>" is.
- L) We should also try to consistently choose certain phraseology to refer to the various data types. For example, the SQL standard uses "character string type", "character data type", "<character string type>", "data type CHARACTER", and several other phrases to describe what is probably the same thing. In general, unless you *really* mean the specific <key word> CHARACTER, you should probably choose "character string type". An analogous guideline applies to the other <data type>s, of course - which highlights another issue: consistency in the use of "data type" vs. "<data type>". I recommend the less formal "data type" unless you really mean the BNF non terminal "<data type>", which is probably appropriate only when discussing the syntax.

#### 6.1.4 Use of Notes

Change proposals often must draw the (proposal) reader's attention to some detail or explanation. This is best done by means of a note starting off with "**Note to proposal reader:**".

Similarly, change proposal sometimes need to call the Editor's attention to some detail, such as the need to assign an SQLSTATE class code or subclass code to a condition. This is best done by means of a note starting off with "**Note to the Editor:**".

However, there is also the need for change proposals to insert notes into the text of the document that they modify, so that the reader of the resulting standard has access to the note text. This is best done by specifying something like "NOTE nnn: text of the note".

#### 6.1.5 Check list

All changes to the document must be specified explicitly. It is unfair to the Editor to expect him to make changes left implicit or unspecified.

The following list of clauses and annex the need review is provided in the hope of reducing the number of incomplete change proposals:

- Conformance Clause,
- Status Codes Clause,
- Annex A, "Implementation-defined elements",
- Annex B, "Implementation-dependent elements",

- A proposal that solves a Possible Problem, Wordsmithing or Still Image Opportunity, whether intentionally or not, should say so prominently, so that the Editor's Notes may be kept up-to-date.

A proposal that introduces a Possible Problem, for example because it is admittedly incomplete should specify an appropriate addition to the Possible Problems list. The same applies to language opportunities.

#### 6.1.5.1 A Checklist of Issues to be Addressed by Change Proposals

- 1) **Concepts** — If a proposal introduces new concepts into the SQL/MM or substantially modifies support for existing concepts, then Clause 4, "Concepts" must be updated to reflect the new reality.
- 2) **Static methods** — For new or modified user-defined types, define appropriate static methods.
- 3) **Constructor methods** — For new or modified user-defined instantiable types, define appropriate SQL-invoked constructor methods to be used with the NEW operator.
- 4) **Cast definitions** — For new or modified user-defined types, define appropriate cast definitions (CREATE CAST).
- 5) **Ordering definitions** — For new or modified user-defined types, define appropriate cast definitions (CREATE ORDERING).
- 6) **SQL Transforms definitions** — For new or modified user-defined types, define appropriate cast definitions (CREATE TRANSFORM).
- 7) **Conformance Clause** — Every new type and routine, however small, must be dealt with in the Conformance Clause.
- 8) **New Status Codes** — Whenever any new exception condition or completion condition is used in the document, it must be accompanied by a specification to include it in the SQLSTATE value tables.
- 9) **Closing Possible Problems** — Many proposals resolve previously-discovered problems in the Still Image document or fulfill a previously-stated desire for a new feature. Proposals must state whether their acceptance can result in closing existing Possible Problems or Language Opportunities.
- 10) **Any new Possible Problems** — If a proposal identifies, but does not solve, a previously-unidentified (or unrecorded) problem, it must clearly instruct the Editor to enter a new Possible Problem.
- 11) **18-Character identifiers** — All identifiers in the Still Image Document must use 18-Character identifiers so this standard can be implemented on conforming SQL implementations without Feature F391, "Long identifiers".
- 12) **Full data type names for basetypes** — Full data type names for base types must be used. . i.e. CHARACTER instead of CHAR, INTEGER instead of INT, DECIMAL instead of DEC, CHARACTER VARYING instead of VARCHAR, NATIONAL CHARACTER instead of NCHAR, CHARACTER LARGE OBJECT instead of CLOB, and BINARY LARGE OBJECT instead of BLOB.
- 13) **Implementation-defined elements** — Every new specification of some aspect of SQL/MM as implementation-defined must be accompanied by a corresponding entry in the appropriate Annex.
- 14) **Implementation-defined Meta-variables** — Every new specification of some aspect of SQL/MM as implementation-defined meta-variable must be accompanied by a corresponding entry in the appropriate Annex.
- 15) **Implementation-dependent elements** — Every new specification of some aspect of SQL/MM as implementation-dependent must be accompanied by a corresponding entry in the appropriate Annex.

More to be supplied as required.

### 6.1.5.2 Format for Possible Problems and Still Image Opportunities

All Possible Problems and Still Image Opportunities should be provide, preferably in electronic, machine-readable form, to the Editor using the following format:

**Severity:** One of Major Technical, Minor Technical, and Major Editorial.

**Reference:** A specific document and either Clause or Subclause, identified by both number and name.

- The document number should be of the form "Pnn", where "nn" represents the part number, including leading zeros to make it a 2-digit number. For example, P02 would be used for SQL/MM—Part 2: Full-Text. Terminate this with a period to separate it from the Clause or Subclause number.
- The Clause or Subclause number should have two digits per level (including leading zeros) such as "03: or "04.21.01".
- The name should be spelled *exactly* as it is currently specified in the base documents, but *not* enclosed in quotation marks. You should not use a comma between the number and the name.
- The word Clause or Subclause should not be used in this item.
- If the PP or LO doesn't apply to a specific Clause or Subclause in the document, then use the phrase "No specific location".

Examples of properly formatted references are:

- P02.No specific location
- P05.04.06.01 Classes of SQL-statements

**Note at:** A specific location within an identified Clause or Subclause. This would identify the paragraph, Purpose, Definition, Definitional Rule, or Description at which the Editor should put a note referencing the Possible Problem or Still Image Opportunity. If the PP or SO does not apply to such a specific place, then the word "None" should be specified here.

**Source:** The name of the person or the paper number that proposed the PP or SO, as well as the date on which it was submitted.

**Possible Problem:** or **Still Image Opportunity:** This is the actual text of the PP or SO (use only one of those two phrases, of course!), taking as many paragraphs as you need.

**Proposed Solution:** This is the actual text of a proposed solution. If none is proposed, then "None submitted with comment" should be specified.

### 6.1.6 The End Of The Paper

Not everybody's printers and computers work perfectly every time. It sometimes happens that an attempt to print a change proposal or other paper terminates prematurely, but it is not always obvious to the reader that the paper is not completely printed. There are ways to combat this. It is very strongly suggested that you end your change proposals (and, indeed, even discussion papers) with some notation like:

```
=====  
End of paper.  
=====
```

Alternatively, you could choose to number each page with the formula "Page n of m", where "m" is the total number of pages. This approach has the disadvantage that many word processors happily print an incorrect value for "m". If you choose this approach, you will have to be extremely careful that the value is correct. Perhaps you could combine this approach with the previous suggestion for maximum benefit.

## 7 Machine readable change proposals

I would like to thank the ISO participants who have provided me with copies of their change proposals on the archive:

`ftp://jerry.ece.umassd.edu/SC32/WG4/Meetings/AAA_YYYY_MM_Place_Country`

where:

AAA is the three letter meeting location (airport) code,  
YYYY\_MM is the year and month of the meeting, and  
Place\_Country identifies the meeting location.

This practice reduces the editorial workload considerably and makes it much easier to ensure accurate incorporation of the change proposals in the base document. I continue to urge those participants to supply copies of their proposals on the archive. The detailed procedures for posting change proposals are set out in ISO/IEC JTC 1/SC 32/N 00149, *Proposed Guidelines for the Production, Distribution, and Storage of SC32 Documents*.

Alternately, if you are unable to post the change proposal to the archive, then I urge you to bring an electronic form of the proposal on a diskette to the meeting. I will of course either return the diskettes or replace them with blank diskettes on request. I prefer clearly labeled, DOS formatted, 3.5 inch HD diskettes (Double Sided, High Density, 80 Tracks, 135TPI). The preferred file formats are Microsoft Word for Windows 1997 and clear text.

Another alternative may be for you to E-mail your change proposal to my Internet E-mail address:

[Mark.Ashworth@ACM.org](mailto:Mark.Ashworth@ACM.org)

Good luck...

# INTERNATIONAL STANDARD

# ISO/IEC 13249-5

First edition  
2000-??-??

---

## Information technology — Database languages — SQL Multimedia and Application Packages —

### Part 5: Still Image

*Technologies de l'information – Langues de bases de données –  
Multimédia SQL et paquetages d'application –*

*Partie 5:*

Document type: International Standard  
Document subtype: Not applicable  
Document stage: **(20) Preparatory**  
Document language: E



Reference Number  
ISO/IEC 13249-5:2000(E)

© ISO/IEC 2000 - All rights reserved

Printed on: October 23, 2000 19:17 EST / Version 6

**PDF disclaimer**

This PDF file may contain embedded typefaces. In accordance with Adobe's licensing policy, this file may be printed or viewed but shall not be edited unless the typefaces which are embedded are licensed to and installed on the computer performing the editing. In downloading this file, parties accept therein the responsibility of not infringing Adobe's licensing policy. The ISO Central Secretariat accepts no liability in this area.

Adobe is a trademark of Adobe Systems Incorporated.

Details of the software products used to create this PDF file can be found in the General Info relative to this file; the PDF-creation parameters were optimized for printing. Every care has been taken to ensure that the file is suitable for use by ISO member bodies. In the unlikely event that a problem relating to it is found, please inform the Central Secretariat at the address given below.

© ISO/IEC 2000

All rights reserved. Unless otherwise specified, no part of this publication may be reproduced or utilized in any form or by any means, electronic or mechanical, including photocopying and microfilm, without permission in writing from either ISO at the address below or ISO's member body in the country of the requester.

ISO copyright office  
Case postale 56 • CH-1211 Geneva 20 • Switzerland  
Tel. + 41 22 749 01 11  
Fax +41 22 734 10 79  
E-mail [copyright@iso.ch](mailto:copyright@iso.ch)  
Web [www.iso.ch](http://www.iso.ch)

<b>Contents</b>	<b>Page</b>
Foreword .....	vii
Introduction.....	viii
<b>1</b> <b>Scope</b> .....	<b>1</b>
<b>2</b> <b>Normative references</b> .....	<b>3</b>
2.1 <b>International standards</b> .....	<b>3</b>
<b>3</b> <b>Definitions, notations, and conventions</b> .....	<b>5</b>
3.1 <b>Definitions</b> .....	<b>5</b>
3.1.1 <b>Definitions provided in Part 1</b> .....	<b>5</b>
3.1.2 <b>Definitions provided in Part 5</b> .....	<b>5</b>
3.1.3 <b>Definitions taken from ISO/IEC 9075</b> .....	<b>6</b>
3.1.4 <b>Definitions taken from ISO/IEC 10918-1</b> .....	<b>6</b>
3.2 <b>Notations</b> .....	<b>7</b>
3.2.1 <b>Notations provided in Part 1</b> .....	<b>7</b>
3.2.2 <b>Notations provided in Part 5</b> .....	<b>7</b>
3.3 <b>Conventions</b> .....	<b>8</b>
<b>4</b> <b>Concepts</b> .....	<b>9</b>
4.1 <b>Introduction</b> .....	<b>9</b>
4.2 <b>Concepts taken from ISO/IEC 9075</b> .....	<b>11</b>
4.3 <b>Types representing digital images</b> .....	<b>12</b>
4.3.1 <b>Attributes of the SI_StillImage type</b> .....	<b>12</b>
4.3.2 <b>Methods of the SI_StillImage type</b> .....	<b>12</b>
4.4 <b>Image features</b> .....	<b>13</b>
4.4.1 <b>Feature types</b> .....	<b>13</b>
4.4.2 <b>Assessing the similarity of images</b> .....	<b>14</b>
4.5 <b>Complementary SQL-invoked regular functions</b> .....	<b>15</b>
4.6 <b>Auxiliary type SI_Color</b> .....	<b>17</b>
4.7 <b>The Still Image Information Schema</b> .....	<b>18</b>
<b>5</b> <b>Still Image Types</b> .....	<b>19</b>
5.1 <b>SI_StillImage Types and Routines</b> .....	<b>19</b>
5.1.1 <b>SI_StillImage Type</b> .....	<b>19</b>
5.1.2 <b>SI_StillImage Methods</b> .....	<b>21</b>
5.1.3 <b>SI_setContent Method</b> .....	<b>23</b>
5.1.4 <b>SI_changeFormat Method</b> .....	<b>24</b>
5.1.5 <b>SI_Thumbnail Methods</b> .....	<b>25</b>
5.1.6 <b>Functions Complementing SI_StillImage Methods</b> .....	<b>26</b>
5.1.7 <b>SI_chgContent Function</b> .....	<b>27</b>
5.1.8 <b>SI_convertFormat Function</b> .....	<b>28</b>
5.1.9 <b>SI_getThumbnail Function</b> .....	<b>29</b>
5.1.10 <b>SI_getSizedThmbnl Function</b> .....	<b>30</b>
5.1.11 <b>Functions Complementing Observer Functions of Type SI_StillImage</b> .....	<b>31</b>
5.1.12 <b>Functions not intended for Public Use</b> .....	<b>33</b>
<b>6</b> <b>Feature Types</b> .....	<b>39</b>
6.1 <b>SI_AverageColor Type and Routines</b> .....	<b>39</b>
6.1.1 <b>SI_AverageColor Type</b> .....	<b>39</b>
6.1.2 <b>SI_AverageColor Methods</b> .....	<b>41</b>
6.1.3 <b>SI_Score Method</b> .....	<b>43</b>
6.1.4 <b>SI_fndAverageColor Function</b> .....	<b>44</b>
6.1.5 <b>SI_mkAverageColor Function</b> .....	<b>45</b>

6.1.6	SI_ScoreByAvrgClr Function.....	46
6.2	SI_ColorHistogram Type and Routines.....	47
6.2.1	SI_ColorHistogram Type.....	47
6.2.2	SI_ColorHistogram Methods.....	49
6.2.3	SI_Append Method.....	52
6.2.4	SI_Score Method.....	53
6.2.5	SI_findColorHstgrm Function.....	54
6.2.6	SI_mkColorHstgrm Function.....	55
6.2.7	SI_arrayClrHstgrm Function.....	56
6.2.8	SI_appendClrHstgrm Function.....	57
6.2.9	SI_ScoreByClrHstgr Function.....	58
6.3	SI_PositionalColor Type and Routines.....	59
6.3.1	SI_PositionalColor Type.....	59
6.3.2	SI_PositionalColor Method.....	60
6.3.3	SI_Score Method.....	61
6.3.4	SI_findPositColor Function.....	62
6.3.5	SI_ScoreByPositClr Function.....	63
6.4	SI_Texture Type and Routines.....	64
6.4.1	SI_Texture Type.....	64
6.4.2	SI_Texture Method.....	65
6.4.3	SI_Score Method.....	66
6.4.4	SI_findTexture Function.....	67
6.4.5	SI_ScoreByTexture Function.....	68
6.5	SI_FeatureList Type and Routines.....	69
6.5.1	SI_FeatureList Type.....	69
6.5.2	SI_FeatureList Method.....	72
6.5.3	SI_setFeature Methods.....	74
6.5.4	SI_Score Method.....	77
6.5.5	SI_mkFeatureList Function.....	79
6.5.6	SI_ScoreByFtrList Function.....	80
6.5.7	Regular Functions Complementing SI_setFeature Methods.....	81
6.5.8	Regular Functions Complementing Observer Functions of type SI_FeatureList.....	83
6.6	Auxiliary Types and Routines.....	85
6.6.1	SI_Color Type.....	85
6.6.2	SI_RGBColor Method.....	86
6.6.3	SI_mkRGBColor Function.....	87
7	SQL/MM Still Image Information Schema.....	89
7.1	Introduction.....	89
7.2	SI_IMAGE_FORMATS view.....	90
7.3	SI_IMAGE_FORMAT_CONVERSIONS view.....	91
7.4	SI_IMAGE_FORMAT_FEATURES view.....	92
7.5	SI_THUMBNAIL_FORMATS view.....	93
7.6	SI_VALUES view.....	94
7.7	Short name views.....	95
8	SQL/MM Still Image Definition Schema.....	97
8.1	Introduction.....	97
8.2	SI_IMAGE_FORMATS base table.....	98
8.3	SI_IMAGE_FORMAT_CONVERSIONS base table.....	99
8.4	SI_IMAGE_FORMAT_FEATURES base table.....	100
8.5	SI_THUMBNAIL_FORMATS base table.....	101
8.6	SI_VALUES base table.....	102
9	Status Codes.....	103
10	Conformance.....	105
10.1	Requirements for conformance.....	105
10.2	Claims of conformance.....	108
	Annex A.....	109

<b>A.1</b>	<b>Implementation-defined Meta-variables</b> .....	<b>110</b>
<b>Annex B</b>	.....	<b>111</b>
<b>B.1</b>	<b>Implementation-dependent Meta-variables</b> .....	<b>113</b>
<b>Index</b>	.....	<b>115</b>

<b>Tables</b>	<b>Page</b>
<b>Table 1 – Method and function name correspondences</b> .....	<b>15</b>
<b>Table 2 – SQLSTATE class and subclass values</b> .....	<b>103</b>

Blank 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 mutual interest. Other international organizations, governmental and non-governmental, in liaison with ISO and IEC, also take part in the work.

International Standards are drafted in accordance with the rules given in the ISO/IEC Directives, Part 3.

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

Attention is drawn to the possibility that some elements of this part of ISO/IEC 13249 may be the subject of patent rights. ISO and IEC shall not be held responsible for identifying any or all such patent rights.

International Standard ISO/IEC 13249-5 was prepared by Joint Technical Committee ISO/IEC JTC 1, *Information Technology*, Subcommittee SC32, *Data Management and interchange*.

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*

Annexes A and B of this part of ISO/IEC 13249 are for information only.

## 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 implementor 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.

In the text of this part of ISO/IEC 13249, Clauses begin a new odd-numbered page, and in Clause 5, "Still Image Types", through Clause 10, "Conformance", Subclauses begin a new page. Any resulting blank space is not significant.

## Information Technology — Database Languages — SQL Multimedia and Application Packages — SQL/MM Part 5: Still Image

### 1 Scope

This part of ISO/IEC 13249:

- a) introduces the Still Image part of ISO/IEC 13249,
- b) gives the references necessary for this part of this part of ISO/IEC 13249,
- c) defines notations and conventions specific to this part of this part of ISO/IEC 13249,
- d) defines concepts specific to this part of 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 adhere to the following:

- A still image user-defined type is generic to image data handling. It addresses the need to store, manage and retrieve information based on aspects of image data such as height, width and format and based on image features such as average color, color histogram, positional color and texture.
- 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.

Blank page

## 2 Normative references

The following standards contain provisions that, through reference in this text, constitute provisions of this part of ISO/IEC 13249. For dated references, subsequent amendments to, or revisions of, any of these publications do not apply. However, parties to agreements based on this part of ISO/IEC 13249 are encouraged to investigate the possibility of applying the most recent editions of the normative documents indicated below. For undated references, the latest edition of the normative document referred to applies. Members of IEC and ISO maintain registers of currently valid International Standards.

### 2.1 International standards

ISO/IEC 9075-1:1999, *Information Technology — Database Languages — SQL — Part 1: Framework (SQL/Framework)*.

ISO/IEC 9075-2:1999, *Information Technology — Database Languages — SQL — Part 2: Foundation (SQL/Foundation)*.

ISO/IEC 9075-4:1999, *Information Technology — Database Languages — SQL — Part 4: Persistent Stored Modules (SQL/PSM)*.

ISO/IEC 13249-1:2000, *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*.

Blank page

## **3 Definitions, notations, and conventions**

### **3.1 Definitions**

For the purpose of this part of ISO/IEC 13249, the following definitions apply.

#### **3.1.1 Definitions provided in Part 1**

This part of ISO/IEC 13249 makes use of all terms defined in Part 1 of ISO/IEC 13249.

#### **3.1.2 Definitions provided in Part 5**

This part of ISO/IEC 13249 defines the following terms:

##### **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 digital images in a specific compressed or uncompressed interchange format

##### **3.1.2.5**

##### **image feature**

characteristic (other than inherent image characteristics) of the image data

##### **3.1.2.6**

##### **inherent image characteristics**

image format and particular physical characteristics of a digital image

##### **3.1.2.7**

##### **list of weighted features**

see composite feature

##### **3.1.2.8**

##### **picture element**

see sample in Subclause 3.1.4, "Definitions taken from ISO/IEC 10918-1"

##### **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 Definitions

#### 3.1.2.11

##### **thumbnail**

a raw image which was obtained from another raw image by downsizing

#### 3.1.3 Definitions taken from ISO/IEC 9075

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

#### 3.1.4 Definitions taken from ISO/IEC 10918-1

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

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 phases "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

## **3.2 Notations**

### **3.2.1 Notations provided in Part 1**

The notations used in this part of ISO/IEC 13249 are defined in Part 1 of ISO/IEC 13249.

### **3.2.2 Notations provided in Part 5**

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 Part 1 of ISO/IEC 13249.

## 4 Concepts

### 4.1 Introduction

In the context of this part of ISO/IEC 13249, digital images are effectively 2-dimensional arrays of picture elements, or samples. The internal representation of a sample in the raw image itself is image format specific. The color value of a sample in the image data might be represented by an index in a color look up table, spread over multiple color planes, multiple binary strings which represent the single components of a color, or in any other image format specific way.

An image format is a set of conventions for storing digital images in an interchange format. An image consists of the representation and organization of data that constitute the picture elements, and prescriptions about auxiliary data that control the interpretation and processing of the digital image information according to that format.

A color space, which is used to represent the color values of the samples, is either defined 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 columns of the image data;
- the height of the raw image is the number of lines 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 and features from the raw image.

NOTE 4 Features are described in Subclause 4.4, "Image features".

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;
  - 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 or the inherent image characteristics of the image;
  - obtaining a thumbnail from an *SI\_StillImage* value.
- Feature data types that abstract from certain characteristics of the pictorial information contained in images; these data types provide facilities for:

#### 4.1 Introduction

- 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 supports Core SQL as defined by ISO/IEC 9075. 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

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 digital images

#### 4.3.1 Attributes of the *SI\_StillImage* type

The *SI\_StillImage* type is an abstraction for digital 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\_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.

#### 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 information when *SI\_StillImage* values are to be constructed from a raw image whose format is not a supported one;
- *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\_Thumbnail*: obtains a thumbnail from an *SI\_StillImage* 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\_format*: returns the format indication of the image;
- *SI\_height*: returns the number of lines of the image;
- *SI\_width*: returns the number of columns 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 samples of 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  $m$  average 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\_fndAverageColor*;
- Color histogram feature: *SI\_findColorHstgrm*;
- Positional color feature: *SI\_findPositColor*;
- Texture feature: *SI\_findTexture*.

#### 4.4 Image features

*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 must 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_setContent	BINARY LARGE OBJECT	SI_chgContent
SI_StillImage	SI_changeFormat	CHARACTER VARYING	SI_convertFormat
SI_StillImage	SI_content		SI_getContent
SI_StillImage	SI_contentLength		SI_getContentLngh
SI_StillImage	SI_format		SI_getFormat
SI_StillImage	SI_height		SI_getHeight
SI_StillImage	SI_width		SI_getWidth
SI_StillImage	SI_Thumbnail		SI_getThumbnail
SI_StillImage	SI_Thumbnail	INTEGER, INTEGER	SI_getSizedThmbnl
SI_AverageColor	SI_AverageColor	SI_StillImage	SI_fndAverageColor
SI_AverageColor	SI_AverageColor	SI_Color	SI_mkAverageColor
SI_AverageColor	SI_Score	SI_StillImage	SI_ScoreByAvrgClr
SI_ColorHistogram	SI_ColorHistogram	SI_StillImage	SI_findColorHstgrm
SI_ColorHistogram	SI_ColorHistogram	SI_Color, DOUBLE PRECISION	SI_mkColorHistogram
SI_ColorHistogram	SI_ColorHistogram	SI_Color ARRAY, DOUBLE PRECISION ARRAY	SI_arrayClrHstgrm
SI_ColorHistogram	SI_Append	SI_Color, DOUBLE PRECISION	SI_appendClrHstgrm
SI_ColorHistogram	SI_Score	SI_StillImage	SI_ScoreByClrHstgr
SI_PositionalColor	SI_PositionalColor	SI_StillImage	SI_findPositColor
SI_PositionalColor	SI_Score	SI_StillImage	SI_ScoreByPositClr
SI_Texture	SI_Texture	SI_StillImage	SI_findTexture
SI_Texture	SI_Score	SI_StillImage	SI_ScoreByTexture

## 4.5 Complementary SQL-invoked regular functions

Type Name	Method Name	Parameter Types (if any)	Function Name
SI_FeatureList	SI_FeatureList	SI_AverageColor, DOUBLE PRECISION, SI_ColorHistogram, DOUBLE PRECISION, SI_PositionalColor, DOUBLE PRECISION, SI_Texture, DOUBLE PRECISION	SI_mkFeatureList
SI_FeatureList	SI_setFeature	SI_AverageColor, DOUBLE PRECISION	SI_setAvgClrFtrW
SI_FeatureList	SI_setFeature	SI_ColorHistogram, DOUBLE PRECISION	SI_setClrHstgrFtrW
SI_FeatureList	SI_setFeature	SI_PositionalColor, DOUBLE PRECISION	SI_setPstnlClrFtrW
SI_FeatureList	SI_setFeature	SI_Texture, DOUBLE PRECISION	SI_setTextureFtrW
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, INTEGER, INTEGER	SI_mkRGBColor

#### 4.6 Auxiliary type *SI\_Color*

Color values are encapsulated by the type *SI\_Color*. The 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 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\_IMAGE\_FORMATS that lists the format indications of the supported image formats;
- a view SI\_IMAGE\_FORMAT\_CONVERSIONS that lists pairs of format indications of image formats for which format conversions are supported;
- a view SI\_IMAGE\_FORMAT\_FEATURES that 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 that lists the format indications of image formats from which thumbnails can be derived;
- a view SI\_VALUES that lists implementation-defined 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_format CHARACTER VARYING(SI_MaxFormatLength),
    SI_height INTEGER,
    SI_width INTEGER
)
INSTANTIABLE
NOT FINAL

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

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

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

### 5.1.1 SI\_StillImage Type

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

```
METHOD SI_Thumbnail
  (height INTEGER,
   width INTEGER)
  RETURNS SI_StillImage
  SELF AS RESULT
  LANGUAGE SQL
  DETERMINISTIC
  CONTAINS SQL
  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.

#### 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\_setContent*(*BINARY LARGE OBJECT*),
  - d) a method *SI\_changeFormat*(*CHARACTER VARYING*),
  - e) a method *SI\_Thumbnail*(),
  - f) a method *SI\_Thumbnail*(*INTEGER*, *INTEGER*),
  - g) a function *SI\_chgContent*(*SI\_StillImage*, *BINARY LARGE OBJECT*),
  - h) a function *SI\_convertFormat*(*SI\_StillImage*, *CHARACTER VARYING*),
  - i) a function *SI\_getThumbnail*(*SI\_StillImage*),
  - j) a function *SI\_getSizedThmbnl*(*SI\_StillImage*, *INTEGER*, *INTEGER*).
  - k) a function *SI\_mkStillImage1*(*BINARY LARGE OBJECT*),
  - l) a function *SI\_mkStillImage2*(*BINARY LARGE OBJECT*, *CHARACTER VARYING*),
- 2) The attributes *SI\_content*, *SI\_contentLength*, *SI\_format*, *SI\_height*, and *SI\_width* are read-only. There are no GRANT statements granting EXECUTE privilege on the mutator functions for *SI\_content*, *SI\_contentLength*, *SI\_format*, *SI\_height*, and *SI\_width*.

## 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
  RETURN SELF.
  SI_content(content).
  SI_contentLength(LENGTH(content)).
  SI_format(SI_format(content)).
  SI_height(SI_height(content)).
  SI_width(SI_width(content))

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
    iFormat = eFormat OR
    NOT SI_supportedFormat(eFormat) = 1 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_format(explicitFormat).
  SI_height(SI_height(content)).
  SI_width(SI_width(content));
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*.

ISO/IEC WD 13249-5:2002(E)  
5.1.2 SI\_StillImage Methods

- 2) 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*.
- 3) If any of the following is True for an invocation of the method *SI\_StillImage*(*BINARY LARGE OBJECT*, *CHARACTER VARYING*), an exception condition is raised: *SQL/MM Still Image exception – illegal image format specification*.
  - a) *explicitFormat* is the null value.
  - b) At least one of the following is False:
    - i) *explicitFormat* indicates a supported image format, and *explicitFormat* is equivalent to the format derived from *content*.
    - ii) *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* ).

### 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 SI_format(content) <> SELF.SI_format THEN
      SIGNAL InvalidInput
        SET MESSAGE_TEXT = 'incorrect image format';
    END IF;
    RETURN SELF.
      SI_content(content).
      SI_contentLength(LENGTH(content)).
      SI_height(SI_height(content)).
      SI_width(SI_width(content));
  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*.

### 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 localContent BINARY LARGE OBJECT(SI_MaxContentLength);

    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 SELF.
            SI_content(localContent).
            SI_contentLength(LENGTH(localContent)).
            SI_format(targetFormat).
            SI_height(SI_height(localContent)).
            SI_width(SI_width(localContent));
    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 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*.
  - b) If *targetFormat* and *SELF.SI\_format* are equivalent, *SELF* is returned unchanged.
  - c) Otherwise, *SELF.SI\_content* is converted to *targetFormat*, and the values of *SELF.SI\_contentLength*, *SELF.SI\_format*, *SELF.SI\_height*, and *SELF.SI\_width* are adjusted to reflect the new value of *SELF.SI\_content*.

### 5.1.5 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 NEW 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_width OR
       NOT SI_supportedThumbnail(SELF.SI_format) 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()*.
  - a) This method takes no input parameters.
  - b) The values for the attributes *SI\_height* and *SI\_width* for the resulting thumbnail are implementation-dependent.
- 2) The method *SI\_Thumbnail(INTEGER, INTEGER)*.
  - a) takes the following input parameters:
    - i) an INTEGER value *height*,
    - ii) an INTEGER value *width*.
  - b) 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*.

## 5.1.6 Functions Complementing SI\_StillImage Methods

## 5.1.6 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
  CALLED 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
  CONTAINS SQL
  CALLED ON NULL INPUT
  RETURN NEW SI_StillImage(content, explicitFormat)
```

## 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*,

### 5.1.7 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
CALLED 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*.

### 5.1.8 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
  CONTAINS SQL
  CALLED 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*.

### 5.1.9 SI\_getThumbnail Function

#### Purpose

The *SI\_getThumbnail* derives a thumbnail from an *SI\_StillImage* value.

#### Definition

```
CREATE FUNCTION SI_getThumbnail  
  (image SI_StillImage)  
  RETURNS SI_StillImage  
  LANGUAGE SQL  
  DETERMINISTIC  
  CONTAINS SQL  
  RETURNS NULL ON NULL INPUT  
  STATIC DISPATCH  
  RETURN image.SI_Thumbnail()
```

#### Description

- 1) The function *SI\_getThumbnail(SI\_StillImage)* takes the following input parameter:
  - a) an *SI\_StillImage* value *image*.

## 5.1.10 SI\_getSizedThmbnl Function

### 5.1.10 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
  CONTAINS SQL
  CALLED 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*.

### 5.1.11 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_getContentLength
  (image SI_StillImage)
  RETURNS INTEGER
  DETERMINISTIC
  CONTAINS SQL
  RETURNS NULL ON NULL INPUT
  STATIC DISPATCH
  RETURN image.SI_contentLength
```

```
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.

### 5.1.11 Functions Complementing Observer Functions of Type *SI\_StillImage*

#### 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\_getContentLength(SI\_StillImage)* takes the following input parameter:
  - a) an *SI\_StillImage* value *image*.
- 4) *SI\_getContentLength(SI\_StillImage)* returns the length of the binary representation of the *SI\_StillImage* value *image*.
- 5) The function *SI\_getFormat(SI\_StillImage)* takes the following input parameter:
  - a) an *SI\_StillImage* value *image*.
- 6) *SI\_getFormat(SI\_StillImage)* returns the character representation of the image format indication of the *SI\_StillImage* value *image*.
- 7) The function *SI\_getHeight(SI\_StillImage)* takes the following input parameter:
  - a) an *SI\_StillImage* value *image*.
- 8) *SI\_getHeight(SI\_StillImage)* returns the height in samples of the *SI\_StillImage* value *image*.
- 9) The function *SI\_getWidth(SI\_StillImage)* takes the following input parameter:
  - a) an *SI\_StillImage* value *image*.
- 10) *SI\_getWidth(SI\_StillImage)* returns the width in samples of the *SI\_StillImage* value *image*.

## 5.1.12 Functions not intended for Public Use

### Purpose

These functions are only intended to be used within the methods *SI\_StillImage*, *SI\_setContent*, *SI\_changeFormat*, and *SI\_Thumbnail*.

### Definition

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

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

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

## 5.1.12 Functions not intended for Public Use

```

CREATE FUNCTION SI_supportedFeature
  (featureName CHARACTER VARYING(SI_MaxFeatureNameLength),
   sourceImage SI_StillImage)
RETURNS INTEGER
LANGUAGE SQL
DETERMINISTIC
CONTAINS SQL
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
CONTAINS SQL
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
  CONTAINS SQL
  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_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
  CONTAINS SQL
  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\_format*(*BINARY LARGE OBJECT*) determines the image format of its parameter. The result is the null value if the parameter does not contain an image format that is amongst the implementation-defined set of supported image formats. The determination of the image format includes a check of the correctness of the image data itself.
- 2) The function *SI\_height*(*BINARY LARGE OBJECT*) determines and returns the height in pixels of an image.
- 3) The function *SI\_width*(*BINARY LARGE OBJECT*) determines and returns the width in pixels of an image.
- 4) The function *SI\_supportedFeature*(*CHARACTER VARYING*, *SI\_StillImage*):
  - a) This function takes the following input parameters:
    - i) a *CHARACTER VARYING* value *featureName*.
    - ii) an *SI\_StillImage* value *sourceImage*.
  - b) 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*. Otherwise, the result is 0 (zero); i.e. the feature indicated by *featureName* is not supported for *sourceImage*.
- 5) 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, the function *SI\_convert* raises an exception: *SQL/MM Still Image exception – fatal error during image format conversion*.
- 6) The function *SI\_supportedConversion*(*CHARACTER VARYING*, *CHARACTER VARYING*):
  - a) This function takes the following input parameters:
    - i) a *CHARACTER VARYING* value *sourceFormat*,
    - ii) a *CHARACTER VARYING* value *targetFormat*.
  - b) 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*. Otherwise, the result is 0 (zero); i.e., that conversion is not supported.
- 7) The function *SI\_supportedFormat*(*CHARACTER VARYING*):
  - a) This function takes the following input parameter:
    - i) a *CHARACTER VARYING* value *specifiedFormat*.

- b) 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. Otherwise, the result is 0 (zero); i.e. *specifiedFormat* is not a supported image format.
- 8) The private function *SI\_deriveThumbnail(SI\_StillImage)*:
- a) This function takes the following input parameters:
- i) an *SI\_StillImage* value *image*,
  - ii) an INTEGER value *height*,
  - iii) an INTEGER value *width*.
- b) This function derives and returns a thumbnail of its parameter *image*. If a thumbnail can not be derived from *image* then the result *ret* is the NULL value. Otherwise, *ret* returns a thumbnail with the specified *height* and *width*. The format of the thumbnail *ret.SI\_format* is the same as the format of the still image *image*.
- 9) The private function *SI\_supportedThumbnail(CHARACTER VARYING)*:
- a) This function takes the following input parameter:
- i) a CHARACTER VARYING value *sourceFormat*.
- b) 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*. Otherwise, the result is 0 (zero); i.e. a thumbnail can not be derived from an image whose format indication equals *sourceFormat*.

Blank page

## 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
    CONTAINS SQL
    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
    CONTAINS SQL
    RETURNS NULL ON NULL INPUT
```

##### 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\_fndAverageColor(SI\_StillImage)*.
  - e) a function *SI\_mkAverageColor(SI\_Color)*,

6.1.1 *SI\_AverageColor* Type

- f) a function *SI\_ScoreByAvrgClr(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*.
- 3) 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*.

## 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 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 the parameter *sourceImage* of the method *SI\_AverageColor(SI\_StillImage)* or its attribute *sourceImage.SI\_content* is the null value, or if the *SI\_AverageColor* feature is not defined for *sourceImage*, then an exception condition is raised: *SQL/MM Still Image exception – bad input image; average color feature cannot be determined*.
- 3) The method *SI\_AverageColor(SI\_StillImage)* derives an *SI\_AverageColor* value from the parameter *sourceImage*. To that end, each component of all samples is summed separately and the sum of each component is divided by the number of samples to give the values of the components of the resulting color value in the feature value.

6.1.2 SI\_AverageColor Methods

- 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*.

#### 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* or *image.SI\_content* 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\_fndAverageColor Function

### 6.1.4 SI\_fndAverageColor Function

#### Purpose

Return the *SI\_AverageColor* value from an *SI\_StillImage* value.

#### Definition

```
CREATE FUNCTION SI_fndAverageColor
  (sourceImage SI_StillImage)
  RETURNS SI_AverageColor
  DETERMINISTIC
  CONTAINS SQL
  CALLED ON NULL INPUT
  STATIC DISPATCH

  RETURN NEW SI_AverageColor(sourceImage)
```

#### Description

- 1) The function *SI\_fndAverageColor(SI\_StillImage)* takes the following input parameter:
  - a) an *SI\_StillImage* value *sourceImage*.

### 6.1.5 SI\_mkAverageColor Function

#### Purpose

Return a specified *SI\_AverageColor* value.

#### Definition

```
CREATE FUNCTION SI_mkAverageColor  
  (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\_mkAverageColor(SI\_Color)* takes the following input parameter:
  - a) an *SI\_Color* value *averageColor*.

## 6.1.6 SI\_ScoreByAvrgClr Function

### 6.1.6 SI\_ScoreByAvrgClr Function

#### Purpose

Determine and return the score of an *SI\_StillImage* value to a given *SI\_AverageColor* value.

#### Definition

```
CREATE FUNCTION SI_ScoreByAvrgClr
  (feature SI_AverageColor,
   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\_ScoreByAvrgClr(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
CONTAINS SQL
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,

```

ISO/IEC WD 13249-5:2002(E)  
6.2.1 SI\_ColorHistogram Type

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

### Definitional Rules

- 1) The *SI\_MaxHistogramLength* is the implementation-defined maximum number of color/frequency pairs that are admissible in an *SI\_ColorHistogram* feature value.

### 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\_findColorHstgrm(SI\_StillImage)*,
  - g) a function *SI\_mkColorHistogram(SI\_Color, DOUBLE PRECISION)*,
  - h) a function *SI\_arrayClrHstgrm(SI\_Color ARRAY, DOUBLE PRECISION ARRAY)*,
  - i) a function *SI\_appendClrHstgrm(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.
- 3) 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*.

## 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 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

```

## ISO/IEC WD 13249-5:2002(E)

### 6.2.2 SI\_ColorHistogram Methods

```
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 the parameter *sourceImage* of the method *SI\_ColorHistogram(SI\_StillImage)* or its attribute *sourceImage.SI\_content* is the null value, or if the *SI\_ColorHistogram* feature is not defined for *sourceImage*, then an exception condition is raised: *SQL/MM Still Image exception – bad input image; color histogram feature cannot be determined*.
- 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<sub>i</sub>*, each of which is represented by some color *C<sub>i</sub>*. For *i* ranging from 1 (one) to *N*, let *F<sub>i</sub>* be frequency values that are initially all 0 (zero). For each sample of *sourceImage*, *F<sub>i</sub>* is increased by 1 (one) if the color of that sample is one of the colors of *A<sub>i</sub>*. Let *F* be the value that is obtained by summing up all *F<sub>i</sub>* values. The final result is effectively a sequence of pairs (*C<sub>i</sub>*, *F<sub>i</sub> / F*).
- 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
     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;
  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* or *image.SI\_content* 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\_findColorHstgrm Function

### 6.2.5 SI\_findColorHstgrm Function

#### Purpose

Return the *SI\_ColorHistogram* value from an *SI\_StillImage* value.

#### Definition

```
CREATE FUNCTION SI_findColorHstgrm
  (sourceImage SI_StillImage)
  RETURNS SI_ColorHistogram
  DETERMINISTIC
  CONTAINS SQL
  CALLED ON NULL INPUT
  STATIC DISPATCH
  RETURN NEW SI_ColorHistogram(sourceImage)
```

#### Description

- 1) The function *SI\_findColorHstgrm(SI\_StillImage)* takes the following input parameter:
  - a) an *SI\_StillImage* value *sourceImage*.

## 6.2.6 SI\_mkColorHistogram Function

### Purpose

Return a specified *SI\_ColorHistogram* value.

### Definition

```
CREATE FUNCTION SI_mkColorHistogram  
  (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\_mkColorHistogram(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\_arrayClrHstgrm Function

#### Purpose

Return a specified *SI\_ColorHistogram* value.

#### Definition

```
CREATE FUNCTION SI_arrayClrHstgrm
  (colors SI_Color ARRAY[SI_MaxHistogramLength],
   frequencies DOUBLE PRECISION ARRAY[SI_MaxHistogramLength])
RETURNS SI_ColorHistogram
DETERMINISTIC
CONTAINS SQL
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\_arrayClrHstgrm(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\_appendClrHstgrm Function

### Purpose

Extend an *SI\_ColorHistogram* value by another (*SI\_Color*, DOUBLE PRECISION) pair.

### Definition

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

### Description

- 1) The function *SI\_appendClrHstgrm(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

### 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
  CONTAINS SQL
  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_AverageColorPositions
    SI_Color ARRAY[SI_NumberSections]
  )
  INSTANTIABLE
  NOT FINAL

  CONSTRUCTOR METHOD SI_PositionalColor
    (sourceImage SI_StillImage)
    RETURNS SI_PositionalColor
    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) The *SI\_NumberSections* is the implementation-defined number of *SI\_AverageColor* values that are represented by an *SI\_PositionalColor* value.

#### 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\_findPositColor(SI\_StillImage)*,
  - d) a function *SI\_ScoreByPositClr(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\_AverageColorPositions*.
- 3) The attribute *SI\_AverageColorPositions* is not for public use. There are no GRANT statements granting EXECUTE privilege on the observer and mutator functions for the attribute *SI\_AverageColorPositions*.

### 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 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 the parameter *sourceImage* of the method *SI\_PositionalColor(SI\_StillImage)* or its attribute *sourceImage.SI\_content* is the null value, or if the *SI\_PositionalColor* feature is not defined for *sourceImage*, then an exception condition is raised: *SQL/MM Still Image exception – bad input image; positional color feature cannot be determined*.
- 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, and for each rectangle, the average color value is determined. The array, thus computed, of color values which represent average colors is the *SI\_AverageColorPositions* 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 5 The color values representing the average color for each area are determined as described in Description Rule 3) in Subclause 6.1.2, "SI\_AverageColor Methods", for the method *SI\_AverageColor(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 an image, that image is effectively divided into *n* by *m* rectangles, such that the product of *n* and *m* equals *SI\_NumberSections*. The lower the returned value, the better the *n* by *m* average colors of *image* are characterized by the average colors represented by the *SI\_PositionalColor* value used for scoring *image*.

NOTE 6 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* or *image.SI\_content* 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\_findPositColor Function

### 6.3.4 SI\_findPositColor Function

#### Purpose

Return the *SI\_PositionalColor* value from an *SI\_StillImage* value.

#### Definition

```
CREATE FUNCTION SI_findPositColor
  (sourceImage SI_StillImage)
  RETURNS SI_PositionalColor
  DETERMINISTIC
  CONTAINS SQL
  CALLED ON NULL INPUT
  STATIC DISPATCH
  RETURN NEW SI_PositionalColor(sourceImage)
```

#### Description

- 1) The function *SI\_findPositColor(SI\_StillImage)* takes the following input parameter:
  - a) an *SI\_StillImage* value *sourceImage*.

### 6.3.5 SI\_ScoreByPositClr Function

#### Purpose

Determine and return the score of an *SI\_StillImage* value to a given *SI\_PositionalColor* value.

#### Definition

```
CREATE FUNCTION SI_ScoreByPositClr  
  (feature SI_PositionalColor,  
   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\_ScoreByPositClr(SI\_PositionalColor, SI\_StillImage)* takes the following input parameters:
  - a) an *SI\_PositionalColor* value *feature*,
  - b) an *SI\_StillImage* value *image*.

## 6.4.1 SI\_Texture Type

## 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
    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) The *SI\_MaxTextureLength* is the implementation-defined number of bytes needed for the implementation-defined encoded representation of an *SI\_Texture*.

## 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.
- 3) 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*.

## 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 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 the parameter *sourceImage* of the method *SI\_Texture(SI\_StillImage)* or its attribute *sourceImage.SI\_content* is the null value, or if the *SI\_Texture* feature is not defined for *sourceImage*, then an exception condition is raised: *SQL/MM Still Image exception – bad input image; texture feature cannot be determined*.
- 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.

### 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 that *image*.

Case:

- a) If SELF or *image* or *image.SI\_content* 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  
  CONTAINS SQL  
  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

### 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
  CONTAINS SQL
  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)
SELF AS RESULT
LANGUAGE SQL
DETERMINISTIC
CONTAINS SQL
CALLED ON NULL INPUT,

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

## 6.5.1 SI\_FeatureList Type

```

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

```

```

METHOD SI_setFeature
  (textureFeature SI_Texture,
   textureFeatureWeight DOUBLE PRECISION)
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
CALLED ON NULL INPUT

```

**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\_mkFeatureList(SI\_AverageColor, DOUBLE PRECISION, SI\_ColorHistogram, DOUBLE PRECISION, SI\_PositionalColor, DOUBLE PRECISION, SI\_Texture, DOUBLE PRECISION)*,
  - h) a function *SI\_setAvgClrFtrW(SI\_FeatureList, SI\_AverageColor, DOUBLE PRECISION)*,
  - i) a function *SI\_setClrHstgrFtrW(SI\_FeatureList, SI\_ColorHistogram, DOUBLE PRECISION)*,
  - j) a function *SI\_setPstnlClrFtrW(SI\_FeatureList, SI\_PositionalColor, DOUBLE PRECISION)*,
  - k) a function *SI\_setTextureFtrW(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*.
- 3) 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*.

## 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 (averageColorFeatureWeight IS NULL OR
        averageColorFeatureWeight < 0.0) OR
        (colorHistogramFeatureWeight IS NULL OR
        colorHistogramFeatureWeight < 0.0) OR
        (positionalColorFeatureWeight IS NULL OR
        positionalColorFeatureWeight < 0.0) OR
        (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 *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 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 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 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 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) If the parameter *averageColorFeatureWeight* is the null value, or if it is less than 0 (zero), then an exception condition is raised: *SQL/MM Still Image exception – incorrect feature list specification*.
- 3) 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*.
- 4) The method *SI\_setFeature(SI\_ColorHistogram, DOUBLE PRECISION)* takes the following input parameters:
  - a) an *SI\_ColorHistogram* value *colorHistogramFeature*,

6.5.3 SI\_setFeature Methods

- b) a DOUBLE PRECISION value *colorHistogramFeatureWeight*.
- 5) If the parameter *colorHistogramFeatureWeight* is the null value, or if it is less than 0 (zero), then an exception condition is raised: *SQL/MM Still Image exception – incorrect feature list specification*.
- 6) 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*.
- 7) 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*.
- 8) If the parameter *positionalColorFeatureWeight* is the null value, or if it is less than 0 (zero), then an exception condition is raised: *SQL/MM Still Image exception – incorrect feature list specification*.
- 9) 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*.
- 10) 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*.
- 11) If the parameter *textureFeatureWeight* is the null value, or if it is less than 0 (zero), then an exception condition is raised: *SQL/MM Still Image exception – incorrect feature list specification*.
- 12) 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 result DOUBLE PRECISION;

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

## 6.5.4 SI\_Score Method

- 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:

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

### 6.5.5 SI\_mkFeatureList Function

#### Purpose

Return a specified *SI\_FeatureList* value.

#### Definition

```
CREATE FUNCTION SI_mkFeatureList
  (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\_mkFeatureList(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

### 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
  CALLED 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_setAvgClrFtrW
  (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_setClrHstgrFtrW
  (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_setPstnlClrFtrW
  (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_setTextureFtrW
  (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)

```

## 6.5.7 Regular Functions Complementing SI\_setFeature Methods

## Description

- 1) The function *SI\_setAvgClrFtrW*(*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\_setClrHstgrFtrW*(*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*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

```

6.5.8 Regular Functions Complementing Observer Functions of type `SI_FeatureList`

```

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
```

#### Description

1) The *SI\_Color* type provides for public use:

- a) a method *SI\_RGBColor(INTEGER, INTEGER, INTEGER)*,
- b) a function *SI\_mkRGBColor(INTEGER, INTEGER, INTEGER)*.

NOTE 7 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) 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.
- 4) 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;
    RETURN SELF.
    --
    -- !! See Description
    --
  END
```

### Definitional Rules

- 1) *SI\_MaxRGBColor* is the implementation-defined maximum value for each component of a color value that is represented 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, is 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 spaces, specified by its red, green, and blue components, is represented by the implementation-dependent set of attributes of the *SI\_Color* type.

### 6.6.3 SI\_mkRGBColor Function

#### Purpose

Construct a specified *SI\_Color* value using the representation of colors in the RGB color space.

#### Definition

```
CREATE FUNCTION SI_mkRGBColor
  (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

The function *SI\_mkRGBColor(INTEGER, INTEGER, INTEGER)* takes the following input parameters:

- a) an INTEGER value *redValue*,
- b) an INTEGER value *greenValue*,
- c) an INTEGER value *blueValue*.

Blank page

## 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\_IMAGE\_FORMATS view

### Function

Identify the supported image formats.

### Definition

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

### 7.3 SI\_IMAGE\_FORMAT\_CONVERSIONS view

#### Function

Identify the source and target image formats for which an image format conversion is supported.

#### Definition

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

## 7.4 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
```

## 7.5 SI\_THUMBNAI\_FORMATS view

### Purpose

Identify the image formats from which thumbnails can be derived.

### Definition

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

## 7.6 SI\_VALUES view

### Purpose

Identify the implementation-defined values.

### Definition

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

## 7.7 Short name views

### Function

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_SOURCE_FORMAT, SI_TARGET_FORMAT
  FROM SI_DEFINITION_SCHEMA.SI_IMAGE_FORMAT_FEATURES

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

Blank page

## 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\_IMAGE\_FORMATS base table

### Function

Identify the supported image formats.

### Definition

```
CREATE TABLE SI_IMAGE_FORMATS
(
  SI_FORMAT CHARACTER VARYING(SI_MaxFormatLength) NOT NULL,
  CONSTRAINT SI_FORMATS_PRIMARY_KEY PRIMARY KEY(SI_FORMAT)
)
```

### 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.

### 8.3 SI\_IMAGE\_FORMAT\_CONVERSIONS base table

#### Function

Identify the source and target image formats for which an image format conversion is supported.

#### Definition

```
CREATE TABLE SI_IMAGE_FORMAT_CONVERSIONS
(
  SI_SOURCE_FORMAT CHARACTER VARYING(SI_MaxFormatLength) NOT NULL,
  SI_TARGET_FORMAT CHARACTER VARYING(SI_MaxFormatLength) NOT NULL,

  CONSTRAINT SI_CONVERSION_PRIMARY_KEY
    PRIMARY KEY(SI_SOURCE_FORMAT, SI_TARGET_FORMAT),
  CONSTRAINT SI_SOURCE_FORMAT_SUPPORTED FOREIGN KEY ( SI_SOURCE_FORMAT )
    REFERENCES SI_IMAGE_FORMATS ( SI_FORMAT ),
  CONSTRAINT SI_TARGET_FORMAT_SUPPORTED FOREIGN KEY ( SI_TARGET_FORMAT )
    REFERENCES SI_IMAGE_FORMATS ( SI_FORMAT )
)
```

#### Definitional Rules

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

#### Description

- 1) The values of *SI\_SOURCE\_FORMAT* and *SI\_TARGET\_FORMAT* are character representations of image formats that are supported by an implementation.
- 2) An *SI\_StillImage* value whose *SI\_format* value is *SI\_SOURCE\_FORMAT* can be converted into an *SI\_StillImage* value whose *SI\_format* value is *SI\_TARGET\_FORMAT*.

## 8.4 SI\_IMAGE\_FORMAT\_FEATURES base table

## 8.4 SI\_IMAGE\_FORMAT\_FEATURES base table

## Purpose

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

## Definition

```
CREATE TABLE SI_IMAGE_FORMAT_FEATURES
(
  SI_FORMAT CHARACTER VARYING(SI_MaxFormatLength) NOT NULL,
  SI_FEATURE_NAME CHARACTER VARYING(SI_MaxFeatureNameLength) NOT NULL,

  CONSTRAINT SI_FORMAT_FEATURES_PRIMARY_KEY
    PRIMARY KEY(SI_FORMAT, SI_FEATURE_NAME),
  CONSTRAINT SI_FORMAT_FEATURES_SUPPORT_FMT FOREIGN KEY (SI_FORMAT )
    REFERENCES SI_IMAGE_FORMATS (SI_FORMAT ),
  CONSTRAINT SI_FORMAT_FEATURE_NAMES
    CHECK(SI_FEATURE_NAME IN ('SI_AverageColor', 'SI_ColorHistogram',
    'SI_PositionalColor', 'SI_Texture'))
)
```

## Definitional Rules

- 1) *SI\_MaxFormatLength* is the implementation-defined maximum length for the character representation of an image format indication.
- 2) *SI\_MaxFeatureNameLength* is the implementation-defined maximum length for the character representation of a basic feature name.

## Description

- 1) The values of *SI\_FORMAT* are character representations of image formats that are supported by an implementation and for which a certain basic feature is supported.
- 2) The values of *SI\_FORMAT\_FEATURES* are character representations of the names of the corresponding basic features.

## 8.5 SI\_THUMBNAIL\_FORMATS base table

### Purpose

Identify the image formats from which thumbnails can be derived.

### Definition

```
CREATE TABLE SI_THUMBNAIL_FORMATS
(
  SI_FORMAT CHARACTER VARYING(SI_MaxFormatLength) NOT NULL,

  CONSTRAINT SI_THNL_FORMATS_PRIMARY_KEY PRIMARY KEY (SI_FORMAT),
  CONSTRAINT SI_THNL_FORMATS_SUPPORT_FMT FOREIGN KEY (SI_FORMAT)
    REFERENCES SI_IMAGE_FORMATS (SI_FORMAT)
)
```

### Definitional Rules

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

### Description

- 1) The values of *SI\_FORMAT* are character representations of image formats that are both supported by an implementation and from which thumbnails can be derived.

## 8.6 SI\_VALUES base table

### Purpose

Identify the implementation-defined values.

### Definition

```
CREATE TABLE SI_VALUES
(
  SI_VALUE CHARACTER VARYING(SI_MaxValueLength) NOT NULL,
  CONSTRAINT SI_VALUES_PRIMARY_KEY PRIMARY KEY (SI_VALUE)
)
```

### Definitional Rules

- 1) *SI\_MaxValueLength* is the implementation-dependent maximum length for the character representation of an implementation-defined value.

### Description

- 1) The values of *SI\_VALUE* are character representations of the implementation-defined values.

## 9 Status Codes

The character string value returned in an SQLSTATE parameter comprises a 2-character class value followed by a 3-character subclass value. The class value for each condition and the subclass value or values for each class value are specified in Table 2 – SQLSTATE class and subclass values.

The "Category" column has the following meanings: "S" means that the class value given corresponds to successful completion and is a completion condition; "W" means that the class value given corresponds to a successful completion but with a warning and is a completion condition; "N" means that the class value corresponds to a no-data situation and is a completion condition; "X" means that the class value given corresponds to an exception condition.

For a successful completion code but with a warning, the first two characters of the SQLSTATE are equal to the SQLSTATE condition code class value for *warning* (defined in Subclause 22.1, "SQLSTATE" in Part 2 of ISO/IEC 9075) and third character of the SQLSTATE is 'H'.

**Table 2 – SQLSTATE class and subclass values**

Category	Condition	Class	Subcondition	Subclass
X	data exception	22	null image value	02D
X	SQL/MM Still Image exception	‡	incorrect image format	F01
X	SQL/MM Still Image exception	‡	incorrect average color feature specification	F02
X	SQL/MM Still Image exception	‡	incorrect color histogram feature specification	F03
X	SQL/MM Still Image exception	‡	incorrect feature list specification	F04
X	SQL/MM Still Image exception	‡	incorrect color specification	F05
X	SQL/MM Still Image exception	‡	bad input image; average color feature cannot be determined	F06
X	SQL/MM Still Image exception	‡	bad input image; positional color feature cannot be determined	F07
X	SQL/MM Still Image exception	‡	bad input image; color histogram feature cannot be determined	F08
X	SQL/MM Still Image exception	‡	bad input image; texture feature cannot be determined	F09
X	SQL/MM Still Image exception	‡	illegal image format specification	F10
X	SQL/MM Still Image exception	‡	unsupported image format conversion specified	F11
X	SQL/MM Still Image exception	‡	illegal specification for thumbnail generation	F12
X	SQL/MM Still Image exception	‡	fatal error during image format conversion	F13

‡ If the routine is implemented as an SQL-invoked routine, then the first two characters of the SQLSTATE are equal to the SQLSTATE condition code class for *SQL routine exception* (defined in Subclause 22.1, "SQLSTATE" in ISO/IEC 9075-2).

Otherwise, the routine is implemented as an external routine and the first two characters of the SQLSTATE are equal to the SQLSTATE condition code class for *external routine exception* (defined in Subclause 22.1, "SQLSTATE" in ISO/IEC 9075-2).

Blank page

## 10 Conformance

### 10.1 Requirements for conformance

A conforming implementation shall support one set of the following public user-defined types and routines:

- 1) a set of public user-defined types and associated methods:
  - a) the *SI\_StillImage* user-defined type as defined in Subclause 5.1.1, "SI\_StillImage Type" with the following mandatory methods:
    - i) *SI\_StillImage*(*BINARY LARGE OBJECT*),
    - ii) *SI\_StillImage*(*BINARY LARGE OBJECT*, *CHARACTER VARYING*),
    - iii) *SI\_setContent*(*BINARY LARGE OBJECT*),
    - iv) *SI\_changeFormat*(*CHARACTER VARYING*),
    - v) *SI\_Thumbnail*(),
    - vi) *SI\_Thumbnail*(*INTEGER*, *INTEGER*),
    - vii) the observer function of the attribute *SI\_content*,
    - viii) the observer function of the attribute *SI\_contentLength*,
    - ix) the observer function of the attribute *SI\_format*,
    - x) the observer function of the attribute *SI\_height*,
    - xi) the observer function of the attribute *SI\_width*.
  - b) the *SI\_AverageColor* user-defined type as defined in Subclause 6.1.1, "SI\_AverageColor Type" with the following mandatory methods:
    - i) *SI\_AverageColor*(*SI\_StillImage*),
    - ii) *SI\_AverageColor*(*INTEGER*, *INTEGER*, *INTEGER*),
    - iii) *SI\_Score*(*SI\_StillImage*).
  - c) the *SI\_ColorHistogram* user-defined type as defined in Subclause 6.2.1, "SI\_ColorHistogram Type" with:
    - i) the following mandatory methods:
      - A) *SI\_ColorHistogram*(*SI\_StillImage*),
      - B) *SI\_ColorHistogram*(*SI\_Color*, *DOUBLE PRECISION*),
      - C) *SI\_Append*(*SI\_Color*, *DOUBLE PRECISION*),
      - D) *SI\_Score*(*SI\_StillImage*).
    - ii) the following optional method:
      - A) *SI\_ColorHistogram*(*SI\_Color ARRAY*, *DOUBLE PRECISION ARRAY*).

- d) the *SI\_PositionalColor* user-defined type as defined in Subclause 6.3.1, "SI\_PositionalColor Type" with the following mandatory methods:
  - i) *SI\_PositionalColor(SI\_StillImage)*,
  - ii) *SI\_Score(SI\_StillImage)*.
- e) the *SI\_Texture* user-defined type as defined in Subclause 6.4.1, "SI\_Texture Type" with the following mandatory methods:
  - i) *SI\_Texture(SI\_StillImage)*,
  - ii) *SI\_Score(SI\_StillImage)*.
- f) the *SI\_FeatureList* user-defined type as defined in Subclause 6.5.1, "SI\_FeatureList Type" with the following mandatory methods:
  - i) *SI\_FeatureList(SI\_AverageColor, DOUBLE PRECISION, SI\_ColorHistogram, DOUBLE PRECISION, SI\_PositionalColor, DOUBLE PRECISION, SI\_Texture, DOUBLE PRECISION)*,
  - ii) *SI\_setFeature(SI\_AverageColor, DOUBLE PRECISION)*,
  - iii) *SI\_setFeature(SI\_ColorHistogram, DOUBLE PRECISION)*,
  - iv) *SI\_setFeature(SI\_PositionalColor, DOUBLE PRECISION)*,
  - v) *SI\_setFeature(SI\_Texture, DOUBLE PRECISION)*,
  - vi) *SI\_Score(SI\_StillImage)*,
  - vii) the observer function of the attribute *SI\_AvgClrFtr*,
  - viii) the observer function of the attribute *SI\_AvgClrFtrWght*,
  - ix) the observer function of the attribute *SI\_ClrHstgrFtr*,
  - x) the observer function of the attribute *SI\_ClrHstgrFtrWght*,
  - xi) the observer function of the attribute *SI\_PstnlClrFtr*,
  - xii) the observer function of the attribute *SI\_PstnlClrFtrWght*,
  - xiii) the observer function of the attribute *SI\_TextureFtr*,
  - xiv) the observer function of the attribute *SI\_TextureFtrWght*.
- g) the *SI\_Color* user-defined type as defined in Subclause 6.6.1, "SI\_Color Type" with the following mandatory method:
  - i) *SI\_RGBColor(INTEGER, INTEGER, INTEGER)*.
- 2) a set of public user-defined types and associated functions:
  - a) the *SI\_StillImage* user-defined type as defined in Subclause 5.1.1, "SI\_StillImage Type" with the following mandatory functions:
    - i) *SI\_mkStillImage1(BINARY LARGE OBJECT)*,
    - ii) *SI\_mkStillImage2(BINARY LARGE OBJECT, CHARACTER VARYING)*,

- iii) *SI\_chgContent(SI\_StillImage, BINARY LARGE OBJECT)*,
  - iv) *SI\_convertFormat(SI\_StillImage, CHARACTER VARYING)*,
  - v) *SI\_getThumbnail(SI\_StillImage)*,
  - vi) *SI\_getSizedThmbnl(SI\_StillImage)*,
  - vii) *SI\_getContent(SI\_StillImage)*,
  - viii) *SI\_getContentLngh(SI\_StillImage)*,
  - ix) *SI\_getFormat(SI\_StillImage)*,
  - x) *SI\_getHeight(SI\_StillImage)*,
  - xi) *SI\_getWidth(SI\_StillImage)*,
- b) the *SI\_AverageColor* user-defined type as defined in Subclause 6.1.1, "SI\_AverageColor Type" with the following mandatory functions:
- i) *SI\_fndAverageColor(SI\_StillImage)*,
  - ii) *SI\_mkAverageColor(SI\_Color)*,
  - iii) *SI\_ScoreByAvrgClr(SI\_AverageColor, SI\_StillImage)*.
- c) the *SI\_ColorHistogram* user-defined type as defined in Subclause 6.2.1, "SI\_ColorHistogram Type" with:
- i) the following mandatory method:
    - A) *SI\_findColorHstgrm(SI\_StillImage)*,
    - B) *SI\_mkColorHistogram(SI\_Color, DOUBLE PRECISION)*,
    - C) *SI\_appendClrHstgrm(SI\_ColorHistogram, SI\_Color, DOUBLE PRECISION)*,
    - D) *SI\_ScoreByClrHstgr(SI\_ColorHistogram, SI\_StillImage)*.
  - ii) the following optional method:
    - A) *SI\_arrayClrHstgrm(SI\_Color ARRAY, DOUBLE PRECISION ARRAY)*.
- d) the *SI\_PositionalColor* user-defined type as defined in Subclause 6.3.1, "SI\_PositionalColor Type" with the following mandatory functions:
- i) *SI\_findPositColor(SI\_StillImage)*,
  - ii) *SI\_ScoreByPositClr(SI\_PositionalColor, SI\_StillImage)*.
- e) the *SI\_Texture* user-defined type as defined in Subclause 6.4.1, "SI\_Texture Type" with the following mandatory functions:
- i) *SI\_findTexture(SI\_StillImage)*,
  - ii) *SI\_ScoreByTexture(SI\_Texture, SI\_StillImage)*.

- f) the *SI\_FeatureList* user-defined type as defined in Subclause 6.5.1, "SI\_FeatureList Type" with the following mandatory functions:
- i) *SI\_mkFeatureList(SI\_AverageColor, DOUBLE PRECISION, SI\_ColorHistogram, DOUBLE PRECISION, SI\_PositionalColor, DOUBLE PRECISION, SI\_Texture, DOUBLE PRECISION)*
  - ii) *SI\_setAvgClrFtrW(SI\_FeatureList, SI\_AverageColor, DOUBLE PRECISION)*,
  - iii) *SI\_setClrHstgrFtrW(SI\_FeatureList, SI\_ColorHistogram, DOUBLE PRECISION)*,
  - iv) *SI\_setPstnlClrFtrW(SI\_FeatureList, SI\_PositionalColor, DOUBLE PRECISION)*,
  - v) *SI\_setTextureFtrW(SI\_FeatureList, SI\_Texture, DOUBLE PRECISION)*,
  - vi) *SI\_getAvgClrFtr(SI\_FeatureList)*,
  - vii) *SI\_getAvgClrFtrW(SI\_FeatureList)*,
  - viii) *SI\_getClrHstgrFtr(SI\_FeatureList)*,
  - ix) *SI\_getClrHstgrFtrW(SI\_FeatureList)*,
  - x) *SI\_getPstnlClrFtr(SI\_FeatureList)*,
  - xi) *SI\_getPstnlClrFtrW(SI\_FeatureList)*,
  - xii) *SI\_getTextureFtr(SI\_FeatureList)*,
  - xiii) *SI\_getTextureFtrW(SI\_FeatureList)*,
  - xiv) *SI\_ScoreByFtrList(SI\_FeatureList, SI\_StillImage)*.
- g) the *SI\_Color* user-defined type as defined in Subclause 6.6.1, "SI\_Color Type" with the following mandatory function:
- i) *SI\_mkRGBColor(INTEGER, INTEGER, INTEGER)*.

A conforming implementation shall also support the views in the schema *SI\_INFORMTN\_SCHEMA* as defined in Clause 7, "SQL/MM Still Image Information Schema".

All other user-defined types and routines defined in this part of ISO/IEC 13249 but not listed above are used only to define the semantics of the public user-defined types and routines. A conforming implementation need not support these additional user-defined types and routines for public use.

## 10.2 Claims of conformance

Claims of conformance to this part of ISO/IEC 13249 shall state:

- 1) The definitions for all elements and actions that this part of ISO/IEC 13049 specifies as implementation-defined.

## Annex A (informative)

### Implementation-defined elements

This Annex references those features that are identified in the body of this part of ISO/IEC 13249 as implementation-defined.

The term implementation-defined is used to identify characteristics that may differ between implementations, but that shall be defined for each particular implementation.

1) Subclause 4.1, "Introduction":

a) Paragraph 4)

A format indication is a character string whose format and content is implementation-defined.

2) Subclause 4.7, "The Still Image Information Schema":

a) List item 5)

a view `SI_VALUES` that lists implementation-defined values.

3) Subclause 5.1.12, "Functions not intended for Public Use":

a) Description 1)

The function `SI_format(BINARY LARGE OBJECT)` determines the image format of its parameter. The result is the null value if the parameter does not contain an image format that is amongst the implementation-defined set of supported image formats.

## A.1 Implementation-defined Meta-variables

- 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.
- 4) The *SI\_MaxHistogramLength* is the implementation-defined maximum number of color/frequency pairs that are admissible in an *SI\_ColorHistogram* feature value.
- 5) *SI\_MaxRGBColor* is the implementation-defined maximum value for each component of a color value that is represented by the RGB color space.
- 6) The *SI\_MaxTextureLength* is the implementation-defined number of bytes needed for the implementation-defined encoded representation of an *SI\_Texture*.
- 7) *SI\_MaxValueLength* is the implementation-dependent maximum length for the character representation of an implementation-defined value.
- 8) The *SI\_NumberSections* is the implementation-defined number of *SI\_AverageColor* values that are represented by an *SI\_PositionalColor* value.

## Annex B (informative)

### Implementation-dependent elements

This Annex references those places where this part of ISO/IEC 13249 states explicitly that the actions of a conforming implementation are implementation-dependent.

The term implementation-dependent is used to identify characteristics that may differ between implementations, but that are not necessarily specified for any particular implementation.

1) Subclause 4.4, "Image features":

a) Paragraph 4)

For a given pair of images, the obtained similarity depends on the kind of feature used for comparison; the exact relationship is implementation-dependent.

2) Subclause 5.1.5, "SI\_Thumbnail Methods":

a) Description 1) b)

The values for the attributes *SI\_height* and *SI\_width* for the resulting thumbnail are implementation-dependent.

3) Subclause 6.1.3, "SI\_Score Method":

a) Description 2) b)

Otherwise, the exact relationship between the values of *SI\_AverageColor*, *SI\_StillImage* and the result of *SI\_Score(SI\_StillImage)* is implementation-dependent.

4) Subclause 6.2.2, "SI\_ColorHistogram Methods":

a) Description 3)

To that end, the color space is divided into an implementation-dependent number *N* of areas *A<sub>i</sub>*, each of which is represented by some color *C<sub>i</sub>*.

5) Subclause 6.2.4, "SI\_Score Method":

a) Description 2) 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) Subclause 6.3.2, "SI\_PositionalColor Method"

a) Description 3)

Further details on the relationship between *sourceImage* and the resulting *SI\_PositionalColor* value, such as the values *n* and *m*, are implementation-dependent.

7) Subclause 6.3.3, "SI\_Score Method":

a) Description 2) NOTE 6)

The way in which *image* is divided into *SI\_NumberSections* of rectangles is implementation-dependent, as well *n* and *m* itself.

b) Description 2) b)

Otherwise, the exact relationship between the values of *SI\_PositionalColor*, *SI\_StillImage*, and the result of *SI\_Score(SI\_StillImage)* is implementation-dependent.

8) Subclause 6.4.1, "SI\_Texture Type":

a) Description 2) a)

The values of which represent image texture characteristics such as coarseness, contrast, and directionality in an implementation-dependent fashion.

9) Subclause 6.4.2, "SI\_Texture Method"

a) Description 3)

The relationship between *sourceImage* and the resulting *SI\_Texture* value is implementation-dependent.

10) Subclause 6.4.3, "SI\_Score Method":

a) Description 2) b)

Otherwise, the exact relationship between the values of *SI\_Texture*, *SI\_StillImage* and the result of *SI\_Score(SI\_StillImage)* is implementation-dependent.

11) Subclause 6.6.1, "SI\_Color Type":

a) Description 2)

The *SI\_Color* type represents color values using an implementation-dependent set of attributes.

b) Description 4)

Values of the *SI\_Color* type represent color values using an implementation-dependent color space.

12) Subclause 6.6.2, "SI\_RGBColor Method":

a) Description 3)

It is implementation-dependent how the color value in the RGB color spaces, specified by its red, green, and blue components, is represented by the implementation-dependent set of attributes of the *SI\_Color* type.

## B.1 Implementation-dependent Meta-variables

- 1) *SI\_MaxValueLength* is the implementation-dependent maximum length for the character representation of an implementation-defined value.
- 2) *SI\_ThumbnailHeight* is the implementation-dependent height for the resulting thumbnail.
- 3) *SI\_ThumbnailWidth* is the implementation-dependent width for the resulting thumbnail.

Blank page

## Index

Index entries appearing in **boldface** indicates a range of pages where a user-defined type is defined in this part of ISO/IEC 13249. All other index entries appear in roman type.

Index page numbers appearing in **boldface** indicates a page or range of pages where the attribute, routine, or user-defined type is specified. Index page numbers appearing in *italics* indicates a page where the word, phrase, attribute, routine, or user-defined type is defined. Index page numbers appearing in roman type indicate a page where the word, phrase, attribute, routine, or type was used.

- A—**
- attribute, 7
- B—**
- base table, 97  
basic image feature, 5  
binary string, 5, 9, 11
- C—**
- color space, 5, 9, 17, 50, 85, 86, 87, 110, 111, 112  
columns, 6, 9, 89  
component, 6, 9, 41, 86, 110, 112  
composite feature, 5, 13  
contrast, 13  
courseness, 13
- D—**
- data type, 1, 9  
digital image, 5, 6, 9, 12  
directionality, 13
- E—**
- EXECUTE privilege, 11, 14, 20, 40, 48, 59, 64, 71, 85
- F—**
- feature, 13  
    average color, 13  
    color histogram, 13  
    positional color, 13  
    texture, 13  
format, 9  
format indication, 9, 12, 37, 109  
function, 11
- H—**
- height, 9
- I—**
- image, 6, 9, 10, 13. *Also see digital image, image data, image feature, image format, raw image, still image, or similarity of images*  
image data, 1, 5, 6, 9, 36  
image feature, 1, 5, 13, 39  
image format, 1, 5, 9, 12, 18, 20, 21, 22, 24, 26, 28, 31, 32, 35, 36, 37, 90, 91, 92, 93, 98, 99, 100, 101, 109, 110  
inherent image characteristics, 5, 9, 12, 13  
interchange format, 5, 6, 9
- L—**
- list of weighted features, 5, 71
- M—**
- mutator function, 12, 14, 20, 40, 48, 59, 64, 71, 85, 105, 106
- N—**
- number of lines, 6, 9
- O—**
- observer function, 14, 40, 48, 59, 64, 85, 105, 106
- P—**
- picture element, 5, 9
- R—**
- raw image, 5, 6, 9, 12, 13
- S—**
- sample, 5, 6, 9, 13, 32, 41, 50  
SELECT privilege, 89  
SI\_Append method. *See SI\_ColorHistogram*  
SI\_appendClrHstgrm function. *See SI\_ColorHistogram*  
SI\_arrayClrHstgrm function. *See SI\_ColorHistogram*  
**SI\_AverageColor, 39–46**  
    SI\_AverageColor method, 15, 39, **41**, 44, 45, 105  
    SI\_AverageColorSpec attribute, 39, 40, 41  
    SI\_fndAverageColor function, 13, 15, 39, **44**, 107  
    SI\_mkAverageColor function, 15, 39, **45**, 107  
    SI\_Score method, 15, 39, **42–43**, 46, 105, 111  
    SI\_ScoreByAvrgClr function, 15, 40, **46**, 107  
    type, 13, 14, 15, 16, **39–40**, 69, 70, 72, 73, 74, 75, 79, 81, 82, 83, 105, 106, 107, 108, 110, 111  
SI\_AverageColor method. *See SI\_AverageColor*  
SI\_AverageColorPositions attribute. *See SI\_PositionalColor*

- SI\_AverageColorSpec attribute. See *SI\_AverageColor*
- SI\_AvgClrFtr attribute. See *SI\_FeatureList*
- SI\_AvgClrFtrWght attribute. See *SI\_FeatureList*
- SI\_changeFormat method. See *SI\_StillImage*
- SI\_chgContent function. See *SI\_StillImage*
- SI\_ClrHstgrFtr attribute. See *SI\_FeatureList*
- SI\_ClrHstgrFtrWght attribute. See *SI\_FeatureList*
- SI\_Color, 85–87**
- SI\_mkRGBColor function, 16, **85**, 108
  - SI\_RGBColor method, 16, **86**, 85, 87, 106
  - type, 15, 16, 39, 40, 41, 42, 45, 47, 48, 49, 50, 51, 52, 55, 56, 57, 59, **85**, 105, 106, 107, 108, 112
- SI\_ColorHistogram, 47–58**
- SI\_Append method, 14, 15, 47, 48, **52**, 57, 105
  - SI\_appendClrHstgrm function, 15, 48, **56–57**, 107
  - SI\_arrayClrHstgrm function, 15, 48, **56**, 107
  - SI\_ColorHistogram method, 15, 47, 48, **49–51**, 54, 55, 56, 105
  - SI\_ColorsList attribute, 47, 48, 49, 50, 52
  - SI\_findColorHstgrm function, 13, 15, 48, **54**, 107
  - SI\_FrequenciesList attribute, 47, 48, 49, 50, 52
  - SI\_mkColorHstgrm function, 15, 48, **55**, 107
  - SI\_Score method, 15, 48, **53**, 58, 105, 111
  - SI\_ScoreByClrHstgr function, 15, 48, **58**, 107
  - type, 13, 14, 15, 16, **47–48**, 69, 70, 72, 73, 74, 75, 79, 81, 82, 83, 105, 106, 107, 108, 110, 111
- SI\_ColorHistogram method. See *SI\_ColorHistogram*
- SI\_ColorsList attribute. See *SI\_ColorHistogram*
- SI\_content attribute. See *SI\_StillImage*
- SI\_contentLength attribute. See *SI\_StillImage*
- SI\_CONVERSION\_PRIMARY\_KEY constraint. See *SI\_IMAGE\_FORMAT\_CONVERSIONS* base table
- SI\_convertFormat function. See *SI\_StillImage*
- SI\_DEFINITION\_SCHEMA schema, 90, 91, 92, 93, 94, 95, 97
- SI\_FeatureList, 69–84**
- SI\_AvgClrFtr attribute, 16, 69, 71, 72, 74, 77, 78, 83, 106
  - SI\_AvgClrFtrWght attribute, 16, 69, 71, 72, 73, 74, 75, 77, 78, 83, 106
  - SI\_ClrHstgrFtr attribute, 16, 69, 71, 72, 74, 75, 77, 78, 83, 106
  - SI\_ClrHstgrFtrWght attribute, 16, 69, 71, 72, 73, 74, 75, 76, 77, 78, 83, 106
  - SI\_FeatureList method, 16, 69, 70, **72–73**, 106
  - SI\_getAvgClrFtr function, 16, 83, 84, 108
  - SI\_getAvgClrFtrW function, 16, 83, 84, 108
  - SI\_getClrHstgrFtr function, 16, 83, 84, 108
  - SI\_getClrHstgrFtrW function, 16, 83, 84, 108
  - SI\_getPstnlClrFtr function, 16, 83, 84, 108
  - SI\_getPstnlClrFtrW function, 16, 83, 84, 108
  - SI\_getTextureFtr function, 16, 84, 108
  - SI\_getTextureFtrW function, 16, 84, 108
  - SI\_mkFeatureList function, 16, 70, **77–79**, 108
  - SI\_PstnlClrFtr attribute, 16, 69, 71, 72, 77, 78, 83, 106
  - SI\_PstnlClrFtrWght attribute, 16, 69, 71, 72, 73, 76, 77, 78, 83, 106
  - SI\_Score method, 16, 70, **77–78**, 80, 106
  - SI\_ScoreByFtrList function, 16, 70, **80**, 108
  - SI\_setAvgClrFtrW function, 16, 70, 81, 82, 108
  - SI\_setClrHstgrFtrW function, 16, 70, 81, 82, 108
  - SI\_setFeature method, 16, 69, 70, **74–76**, 106
  - SI\_setPstnlClrFtrW function, 16, 70, 81, 108
  - SI\_setTextureFtrW function, 16, 70, 81, 108
  - SI\_TextureFtr attribute, 16, 69, 71, 72, 77, 78, 84, 106
  - SI\_TextureFtrWght attribute, 16, 69, 71, 72, 73, 76, 77, 78, 84, 106
  - type, 13, 16, **69–71**, 106, 108
- SI\_FeatureList method. See *SI\_FeatureList*
- SI\_findColorHstgrm function. See *SI\_ColorHistogram*
- SI\_findPositColor function. See *SI\_PositionalColor*
- SI\_findTexture function. See *SI\_Texture*
- SI\_fndAverageColor function. See *SI\_AverageColor*
- SI\_format attribute. See *SI\_StillImage*
- SI\_FORMAT column. See *SI\_IMAGE\_FORMATS* base table, *SI\_IMAGE\_FORMAT\_FEATURES* base table, or *SI\_THUMBNAIL\_FORMATS* base table
- SI\_FORMAT\_CONVRSNS view, **95**
- SI\_FORMAT\_FEATURES\_NAMES constraint. See *SI\_IMAGE\_FORMAT\_FEATURES* base table
- SI\_FORMAT\_FEATURES\_PRIMARY\_KEY constraint. See *SI\_IMAGE\_FORMAT\_FEATURES* base table
- SI\_FORMAT\_FEATURES\_SUPPORT\_FMT constraint. See *SI\_IMAGE\_FORMAT\_FEATURES* base table
- SI\_FORMAT\_NAME column. See *SI\_IMAGE\_FORMAT\_FEATURES* base table
- SI\_FORMATS\_PRIMARY\_KEY constraint. See *SI\_IMAGE\_FORMATS* base table
- SI\_FrequenciesList attribute. See *SI\_ColorHistogram*
- SI\_getSizedThmbnl method. See *SI\_StillImage*
- SI\_getThumbnail method. See *SI\_StillImage*
- SI\_height attribute. See *SI\_StillImage*
- SI\_IMAGE\_FORMAT\_CONVERSIONS base table, 91, 95, **99**
- SI\_CONVERSION\_PRIMARY\_KEY constraint, 99
  - SI\_SOURCE\_FORMAT column, 91, 95, 99
  - SI\_SOURCE\_FORMAT\_SUPPORTED constraint, 99
  - SI\_TARGET\_FORMAT column, 91, 95, 99
  - SI\_TARGET\_FORMAT\_SUPPORTED constraint, 99
- SI\_IMAGE\_FORMAT\_CONVERSIONS view, 18, 34, 36, **91**
- SI\_SOURCE\_FORMAT column, 34
  - SI\_TARGET\_FORMAT column, 34
- SI\_IMAGE\_FORMAT\_FEATURES base table, 92, 95, **100**
- SI\_FEATURE\_NAME column, 92, 100
  - SI\_FORMAT column, 92, 100
  - SI\_FORMAT\_FEATURE\_NAMES constraint, 100
  - SI\_FORMAT\_FEATURES\_PRIMARY\_KEY constraint, 100
  - SI\_FORMAT\_FEATURES\_SUPPORT\_FMT constraint, 100
- SI\_IMAGE\_FORMAT\_FEATURES view, 18, 34, 36, **92**
- SI\_FEATURE\_NAME column, 34, 36
  - SI\_FORMAT column, 34, 36
  - SI\_SOURCE\_FORMAT column, 36
  - SI\_TARGET\_FORMAT column, 36
- SI\_IMAGE\_FORMATS base table, 90, **98**, 99, 100, 101
- SI\_FORMAT column, 90, 98, 99, 100
  - SI\_FORMATS\_PRIMARY\_KEY constraint, 98
- SI\_IMAGE\_FORMATS view, 18, 35, 37, **90**
- SI\_FORMAT column, 35, 37
- SI\_IMAGE\_FRMT\_FTRS view, **95**

- SI\_INFORMTN\_SCHEMA schema, 18, 34, 35, 36, 37, 89, 97, 108
- SI\_MaxContentLength, 19, 20, 21, 23, 24, 26, 27, 31, 33, 34, 35, 110
- SI\_MaxFeatureNameLength, 34, 35, 100, 110
- SI\_MaxFormatLength, 19, 20, 21, 24, 26, 28, 31, 33, 34, 35, 98, 99, 100, 101, 110
- SI\_MaxHistogramLength, 47, 48, 50, 52, 56, 110
- SI\_MaxRGBColor, 86, 110
- SI\_MaxTextureLength, 64, 110
- SI\_MaxValueLength, 102, 110, 113
- SI\_mkAverageColor function. *See SI\_AverageColor*
- SI\_mkColorHistogram function. *See SI\_ColorHistogram*
- SI\_mkFeatureList feature. *See SI\_FeatureList. See SI\_FeatureList*
- SI\_mkRGBColor function. *See SI\_Color*
- SI\_NumberSections, 59, 61, 110, 112
- SI\_PositionalColor, 59–63**
- SI\_AverageColorPositions attribute, 59, 60
  - SI\_findPositColor function, 13, 15, 62, 107
  - SI\_PositionalColor method, 15, 59, 60, 62, 106
  - SI\_Score method, 15, 59, 61, 63, 106, 112
  - SI\_ScoreByPositClr function, 15, 63, 107
  - type, 13, 15, 16, 59, 69, 70, 72, 73, 75, 76, 79, 81, 82, 83, 106, 107, 108, 110, 111, 112
- SI\_PositionalColor method. *See SI\_PositionalColor*
- SI\_PstnlClrFtr attribute. *See SI\_FeatureList*
- SI\_PstnlClrFtrWght attribute. *See SI\_FeatureList*
- SI\_RGBColor method. *See SI\_Color*
- SI\_Score method, 13, 14. *See SI\_AverageColor, SI\_ColorHistogram, SI\_PositionalColor, SI\_Texture, or SI\_FeatureList*
- SI\_ScoreByAvrgClr function. *See SI\_AverageColor*
- SI\_ScoreByClrHstgr function. *See SI\_ColorHistogram*
- SI\_ScoreByPositClr function. *See SI\_PositionalColor*
- SI\_ScoreByTexture function. *See SI\_Texture*
- SI\_setContent method. *See SI\_StillImage*
- SI\_setFeature method. *See SI\_FeatureList*
- SI\_SOURCE\_FORMAT column. *See SI\_IMAGE\_FORMAT\_CONVERSIONS base table*
- SI\_SOURCE\_FORMAT\_SUPPORTED constraint. *See SI\_IMAGE\_FORMAT\_CONVERSIONS base table*
- SI\_StillImage, 19–37**
- SI\_changeFormat method, 15, 19, 20, 24, 28, 105
  - SI\_chgContent function, 15, 26–27, 107
  - SI\_content attribute, 12, 15, 19, 20, 21, 23, 24, 26, 27, 31, 32, 35, 41, 43, 49, 50, 53, 60, 61, 65, 66, 98, 105, 110
  - SI\_contentLength attribute, 12, 15, 19, 20, 21, 23, 24, 31, 105
  - SI\_convert function, 34, 36
  - SI\_convertFormat function, 15, 20, 28, 107
  - SI\_deriveThumbnail function, 25, 35, 37
  - SI\_format attribute, 12, 15, 19, 20, 21, 23, 24, 25, 31, 34, 36, 37, 98, 99, 105
  - SI\_format function, 21, 33, 36, 109
  - SI\_getContent function, 15, 31, 32, 107
  - SI\_getContentLngh function, 15, 31, 32, 107
  - SI\_getFormat function, 15, 31, 32, 107
  - SI\_getHeight function, 15, 31, 32, 107
  - SI\_getSizedThmbnl function, 15, 20, 30, 107
  - SI\_getThumbnail function, 15, 20, 29, 107
  - SI\_getWidth function, 15, 31, 32, 107
  - SI\_height attribute, 12, 15, 19, 20, 21, 23, 24, 25, 31, 105, 111
  - SI\_height function, 21, 33, 36
  - SI\_mkStillImage1 function, 15, 20, 26, 106
  - SI\_mkStillImage2 function, 15, 20, 26, 106
  - SI\_setContent method, 15, 19, 20, 23, 27, 33, 105
  - SI\_SI\_changeFormat method, 33
  - SI\_StillImage method, 12, 15, 19, 20, 21–22, 21, 26, 33, 105
  - SI\_supportedConversion function, 34, 36
  - SI\_supportedFeature function, 34, 36, 41, 49, 60, 65
  - SI\_supportedFormat function, 21, 35, 36
  - SI\_supportedThumbnail function, 25, 35, 37
  - SI\_Thumbnail method, 12, 15, 20, 25, 29, 30, 33, 105
  - SI\_width attribute, 12, 15, 19, 20, 21, 23, 24, 25, 31, 105, 111
  - SI\_width function, 21, 33, 36
  - type, 9, 12, 13, 14, 15, 16, 19–20, 32, 39, 40, 41, 43, 44, 46, 47, 48, 49, 50, 53, 54, 58, 59, 60, 61, 62, 63, 64, 65, 66, 67, 68, 69, 70, 77, 78, 80, 98, 99, 105, 106, 107, 108, 110, 111, 112
- SI\_StillImage method. *See SI\_StillImage*
- SI\_TARGET\_FORMAT\_NAME column. *See SI\_IMAGE\_FORMAT\_CONVERSIONS base table*
- SI\_TARGET\_FORMAT\_SUPPORTED constraint. *See SI\_IMAGE\_FORMAT\_CONVERSIONS base table*
- SI\_Texture, 64–68**
- SI\_findTexture function, 13, 15, 64, 67, 107
  - SI\_Score method, 15, 64, 66, 68, 106, 112
  - SI\_ScoreByTexture function, 15, 64, 68, 107
  - SI\_Texture method, 15, 64, 65, 67, 106
  - SI\_TextureEncoding attribute, 64
  - type, 13, 15, 16, 64, 69, 70, 72, 73, 75, 76, 79, 81, 82, 84, 106, 107, 108, 110, 112
- SI\_Texture method. *See SI\_Texture*
- SI\_TextureEncoding attribute. *See SI\_Texture*
- SI\_TextureFtr attribute. *See SI\_FeatureList*
- SI\_TextureFtrWght attribute. *See SI\_FeatureList*
- SI\_THNL\_FORMATS\_PRIMARY\_KEY constraint. *See SI\_THUMBNAIL\_FORMATS base table*
- SI\_THNL\_FORMATS\_SUPPORT\_FMT constraint. *See SI\_THUMBNAIL\_FORMATS base table*
- SI\_Thumbnail method. *See SI\_StillImage*
- SI\_THUMBNAIL\_FORMATS base table, 93, 95, 101
- SI\_FORMAT column, 93, 95, 101
  - SI\_THNL\_FORMATS\_PRIMARY\_KEY constraint, 101
  - SI\_THNL\_FORMATS\_SUPPORT\_FMT constraint, 101
- SI\_THUMBNAIL\_FORMATS view, 18, 35, 37, 93
- SI\_FORMAT column, 35, 37
- SI\_THUMBNAIL\_FRMTS view, 95
- SI\_VALUE column. *See SI\_VALUES base table*
- SI\_VALUES base table, 94, 102
- SI\_VALUE column, 94, 102
  - SI\_VALUES\_PRIMARY\_KEY constraint, 102
- SI\_VALUES view, 18, 94, 109
- SI\_VALUES\_PRIMARY\_KEY constraint. *See SI\_VALUES base table*
- SI\_width attribute. *See SI\_StillImage*
- similarity of images, 5
- SQL-invoked regular function, 10, 11, 15
- SQLSTATE, 103
- 2202D
    - null image value, 23, 103
  - 2F

SQL routine exception, 103  
 2FF01  
   *incorrect image format*, 23, 103  
 2FF02  
   *incorrect average color feature specification*, 41, 42, 103  
 2FF03  
   *incorrect color histogram feature specification*, 49, 50, 51, 52, 103  
 2FF04  
   *incorrect feature list specification*, 72, 73, 74, 75, 76, 103  
 2FF05  
   *incorrect color specification*, 86, 103  
 2FF06  
   *bad input image; average color feature cannot be determined*, 41, 103  
 2FF07  
   *bad input image; positional color feature cannot be determined*, 60, 103  
 2FF08  
   *bad input image; color histogram feature cannot be determined*, 49, 50, 103  
 2FF09  
   *bad input image; texture feature cannot be determined*, 65, 103  
 2FF10  
   *illegal image format specification*, 21, 22, 103  
 2FF11  
   *unsupported image format conversion specified*, 24, 103  
 2FF12  
   *illegal specification for thumbnail generation*, 25, 103

2FF13  
   *fatal error during image format conversion*, 36, 103  
 38  
   *external routine exception*, 103  
 38F01. See SQLSTATE: 2FF01  
 38F02. See SQLSTATE: 2FF02  
 38F03. See SQLSTATE: 2FF03  
 38F04. See SQLSTATE: 2FF04  
 38F05. See SQLSTATE: 2FF05  
 38F06. See SQLSTATE: 2FF06  
 38F07. See SQLSTATE: 2FF07  
 38F08. See SQLSTATE: 2FF08  
 38F09. See SQLSTATE: 2FF09  
 38F10. See SQLSTATE: 2FF10  
 38F11. See SQLSTATE: 2FF11  
 38F12. See SQLSTATE: 2FF12  
 38F13. See SQLSTATE: 2FF13  
 still image, 1, 6, 9, 14, 19, 37, 39  
 supported format, 9

—T—

thumbnail, 6, 9, 12, 18, 25, 29, 30, 37, 93, 101, 113

—V—

view, 10, 18, 89, 95, 108

—W—

width, 9