Oracle® *inter*Media

Reference

10*g* Release 1 (10.1)

Part No. B10829-01

December 2003

Oracle *inter*Media ("*inter*Media") is a feature that enables Oracle Database to store, manage, and retrieve images, audio, video, or other heterogeneous media data in an integrated fashion with other enterprise information. Oracle *inter*Media extends Oracle Database reliability, availability, and data management to multimedia content in Internet, electronic commerce, and media-rich applications.

ORACLE®

Oracle *inter*Media Reference, 10*g* Release 1 (10.1)

Part No. B10829-01

Primary Author: Ingrid Stuart

Contributor: Rod Ward, Susan Mavris, Melli Annamalai, Todd Rowell, Raja Chatterjee, Robert Abbott, Albert Landeck, Vishal Rao, Dongbai Guo, Fengting Chen, Manjari Yalavarthy, Joseph Mauro, Rabah Mediouni, Sanjay Agarwal, Bill Voss, Susan Kotsovolos, Rosanne Toohey, Bill Beauregard, Susan Shepard, Helen Grembowicz

Contents

6 ORDImage and ORDImageSignature

7 SQL/MM Still Image

8 ORDVideo

C Video File and Compression Formats

D Image process( ) and processCopy( ) Operators

List of Figures

List of Tables

Send Us Your Comments

Oracle *inter*Media Reference, 10*g* Release 1 (10.1)

Part No. B10829-01

Oracle Corporation welcomes your comments and suggestions on the quality and usefulness of this publication. Your input is an important part of the information used for revision.

- Did you find any errors?
- Is the information clearly presented?
- Do you need more information? If so, where?
- Are the examples correct? Do you need more examples?
- What features did you like most about this manual?

If you find any errors or have any other suggestions for improvement, please indicate the title and part number of the documentation and the chapter, section, and page number (if available). You can send comments to us in the following ways:

- Electronic mail: nedc-doc_us@oracle.com
- FAX: 603.897.3825 Attn: Oracle *inter*Media Documentation
- Postal service:
  Oracle Corporation
  Oracle *inter*Media Documentation
  One Oracle Drive
  Nashua, NH 03062
  USA

If you would like a reply, please give your name, address, telephone number, and electronic mail address (optional).

If you have problems with the software, please contact your local Oracle Support Services.

Preface

This guide describes how to use Oracle *inter*Media *("inter*Media"), which ships with Oracle Database.

For information about Oracle Database and the latest features and options that are available to you, see *Oracle Database New Features*.

Audience

This guide is for application developers and database administrators who are interested in storing, retrieving, and manipulating audio, image, and video data in Oracle Database, including developers of audio, image, and video specialization options. Before using this reference, you should familiarize yourself with the concepts presented in *Oracle interMedia User's Guide*.

If you are interested in only one particular object type, see Chapter 1 for general introductory information, then, for a description of the methods that are common for all object types, refer to Chapter 3. If, for example, you are interested in the ORDImage object type, refer to Chapter 6, the ORDImage reference chapter for a description of the image-specific methods.

For a description of using the relational interface with images, see Chapter 9.

For information on supported image content and compression formats, see Appendix B. For information about using image processing methods, see Appendix D. Finally, for information about the raw pixel image format, see Appendix E.

Documentation Accessibility

Our goal is to make Oracle products, services, and supporting documentation accessible, with good usability, to the disabled community. To that end, our documentation includes features that make information available to users of assistive technology. This documentation is available in HTML format, and contains markup to facilitate access by the disabled community. Standards will continue to evolve over time, and Oracle Corporation is actively engaged with other market-leading technology vendors to address technical obstacles so that our documentation can be accessible to all of our customers. For additional information, visit the Oracle Accessibility Program Web site at

`http://www.oracle.com/accessibility/`

Accessibility of Code Examples in Documentation JAWS, a Windows screen reader, may not always correctly read the code examples in this document. The conventions for writing code require that closing braces should appear on an otherwise empty line; however, JAWS may not always read a line of text that consists solely of a bracket or brace.

Accessibility of Links to External Web Sites in Documentation This documentation may contain links to Web sites of other companies or organizations that Oracle Corporation does not own or control. Oracle Corporation neither evaluates nor makes any representations regarding the accessibility of these Web sites.

Organization

This guide contains the following chapters and appendixes:

Chapter 1, "Introduction to Oracle interMedia"
Introduces multimedia and Oracle *inter*Media.

Chapter 2, "Ensuring Compatibility with Evolving interMedia Object Types"
Provides compatibility information for ensuring future compatibility with evolving object types.

Chapter 3, "Common Methods for interMedia Object Types"
Provides reference information about methods that are common to ORDAudio, ORDDoc, ORDImage, and ORDVideo object types.

Chapter 4, "ORDAudio"

Provides reference information on Oracle *inter*Media ORDAudio object type and methods.

Chapter 5, "ORDDoc"

Provides reference information on Oracle *inter*Media ORDDoc object type and methods.

Chapter 6, "ORDImage and ORDImageSignature"

Provides reference information on Oracle *inter*Media ORDImage object type and methods.

Chapter 7, "SQL/MM Still Image"

Provides reference information on SQL/MM Still Image object types, methods, SQL functions and procedures, and views that identify the supported image formats and implementation-defined values

Chapter 8, "ORDVideo"

Provides reference information on Oracle *inter*Media ORDVideo object type and methods.

Chapter 9, "interMedia Relational Interface Reference"

Provides reference information on Oracle *inter*Media relational interface methods for the ORDAudio, ORDDoc, ORDImage, and ORDVideo object types.

Chapter 10, "ORDSource"

Provides reference information on Oracle *inter*Media ORDSource object type and methods.

Appendix A, "Audio File and Compression Formats"

Describes the supported audio data formats.

Appendix B, "Image File and Compression Formats"

Describes the supported image data formats.

Appendix C, "Video File and Compression Formats"

Describes the supported video data formats.

Appendix D, "Image process( ) and processCopy( ) Operators"
Describes the process( ) and processCopy( ) operators.

Appendix E, "Image Raw Pixel Format"
Describes the raw pixel format.

Appendix F, "Exceptions and Error Messages"
Lists exceptions raised, their causes, and user actions to correct them.

Appendix G, "Deprecated Audio and Video Methods"
Lists the ORDAudio and ORDVideo get methods that were deprecated in release 8.1.6

Related Documentation

Note: For information added after the release of this guide, refer to the online README.txt file in your <*ORACLE_HOME*> directory. Depending on your operating system, this file may be in:

<*ORACLE_HOME*>`/ord/im/admin/README.txt`

Please see your operating system-specific installation guide for more information.

For more information about using *inter*Media in a development environment, see the following documents in the Oracle Database software documentation set:

- *Oracle interMedia User's Guide*
- *Oracle Call Interface Programmer's Guide*
- *Oracle Streams Advanced Queuing User's Guide and Reference*
- *Oracle Database Application Developer's Guide - Large Objects*
- *Oracle Database Concepts*
- *PL/SQL User's Guide and Reference*
- *Oracle interMedia Java Classes Reference*

Oracle error message documentation is available in HTML only. If you have access to the Oracle Database 10*g* Documentation CD-ROM only, you can browse the error

messages by range. Once you find the specific range, use your browser's "find in page" feature to locate the specific message.

When connected to the Internet, you can search for a specific error message using the error message search feature of the Oracle online documentation.

For information about Oracle Locator, see *Oracle Spatial User's Guide and Reference*.

For reference information on both Oracle *inter*Media Java Classes and Oracle *inter*Media Java Classes for Servlets and JSP in Javadoc format, see the Oracle API documentation (also known as Javadoc). The API documentation is available on the Oracle Database 10*g* Documentation CD-ROM and also from the documentation section of the Oracle Technology Network (OTN) Web site at

`http://otn.oracle.com/documentation/`

Note that the information in the reference sections of *Oracle interMedia Java Classes Reference* supersedes the Javadoc documentation.

Many of the examples in this book use the sample schemas of the seed database, which is installed by default when you install Oracle. Refer to *Oracle Database Sample Schemas* for information on how these schemas were created and how you can use them yourself.

Printed documentation is available for sale in the Oracle Store at

`http://oraclestore.oracle.com/`

To download free release notes, installation documentation, white papers, or other collateral, please visit the Oracle Technology Network (OTN). You must register online before using OTN; registration is free and can be done at

`http://otn.oracle.com/membership/`

If you already have a user name and password for OTN, then you can go directly to the documentation section of the OTN Web site at

`http://otn.oracle.com/documentation/`

Conventions

In this guide, Oracle *inter*Media is sometimes referred to as *inter*Media.

In examples, an implied carriage return occurs at the end of each line, unless otherwise noted. You must press the Return key at the end of a line of input.

The following conventions are also used in this guide:

Convention	Meaning
. . .	Vertical ellipsis points in an example mean that information not directly related to the example has been omitted.
. . .	Horizontal ellipsis points in statements or commands mean that parts of the statement or command not directly related to the example have been omitted.
boldface text	Boldface text indicates a term defined in the text.
italic text	Italic text is used for emphasis, book titles, and usr-supplied information.
< >	Angle brackets enclose user-supplied names.
[ ]	Brackets enclose optional clauses from which you can choose one or none.

Changes to This Guide

The following substantive changes have been made to this guide since release 9.0.1:

- Documentation for new image objects that comply with the first edition of the ISO/IEC 13249-5:2001 SQL MM Part5:StillImage standard (commonly referred to as the SQL/MM Still Image standard) is added. See Chapter 7.
- The following information has been moved to *Oracle interMedia User's Guide*:
  - Extending Oracle *inter*Media (previously in Chapter 1)
  - *inter*Media architecture (previously in Chapter 1)
  - Content-based retrieval concepts (previously Chapter 2)
  - *inter*Media examples (previously Chapter 3)
  - Tuning tips for the DBA (previously Chapter 11)
  - Sample programs (previously Appendix F)
  - Packages or PL/SQL plug-ins (previously included at the end of the chapters that describe ORDAudio, ORDDoc, ORDVideo, and ORDSource)

1

Introduction to Oracle *inter*Media

Oracle *inter*Media ("*inter*Media") is a feature that enables Oracle Database to store, manage, and retrieve images, audio, video, or other heterogeneous media data in an integrated fashion with other enterprise information. Oracle *inter*Media extends Oracle Database reliability, availability, and data management to multimedia content in traditional, Internet, electronic commerce, and media-rich applications.

See *Oracle interMedia User's Guide* for conceptual information and information on application development.

1.1 Multimedia Object Types and Methods

Oracle *inter*Media provides the ORDAudio, ORDDoc, ORDImage, ORDImageSignature, ORDVideo, and SI_StillImage object types and methods for:

- Extracting metadata and attributes from multimedia data
- Getting and managing multimedia data from Oracle *inter*Media, Web servers, file systems, and other servers
- Performing manipulation operations on image data

1.2 Multimedia Storage

Oracle *inter*Media provides the ORDSource object type and methods for multimedia data source manipulation. The ORDAudio, ORDDoc, ORDImage, and ORDVideo object types all contain an attribute of type ORDSource.

Note: ORDSource methods should not be called directly. Instead, invoke the wrapper method of the media object corresponding to the ORDSource method. ORDSource method information is presented only for users who want to write their own user-defined sources.

2

Ensuring Compatibility with Evolving *inter*Media Object Types

The Oracle *inter*Media ("*inter*Media") object types may evolve by adding new object attributes in a future release. Client-side applications that want to maintain compatibility with the current release of the *inter*Media object types (ORDAudio, ORDImage, ORDVideo, and ORDDoc), even after a server upgrade that includes evolved object types, are advised to do the following:

- Make a call to the compatibility initialization function at the beginning of the application, if necessary (see Section 2.1).
- Use the static constructor functions, init( ), in INSERT statements that are provided beginning with release 8.1.7 (see the sections on ORDAudio Constructors on page 4-8, ORDDoc Constructors on page 5-6, ORDImage Constructors on page 6-7, and ORDVideo Constructors on page 8-8). Do not use the default constructors because INSERT statements using the default constructors will fail if the *inter*Media object types have added new attributes.

> **Note:** If you do not take the preceding recommended actions, you may have to upgrade your client and perhaps even edit and recompile your application when you upgrade to a newer database release with evolved *inter*Media object types.

The information in this chapter is not applicable to the SQL/MM Still Image object types.

2.1 When and How to Call the Compatibility Initialization Function

Only client-side applications that statically recognize the structure of the *inter*Media object types need to make a call to the compatibility initialization function. Server-side stored procedures will automatically use the newly installed (potentially changed) *inter*Media object types after an upgrade, so you do not need to call the compatibility initialization function from server-side stored procedures.

Client-side applications that do not statically (at compile time) recognize the structure of *inter*Media object types do not need to call the compatibility initialization function. OCI applications that determine the structure of the *inter*Media object types at runtime, through the OCIDescribeAny call, do not need to call the compatibility initialization function.

Client-side applications written in OCI that have been compiled with the C structure of *inter*Media object types (generated by Oracle type translator) should make a call to the server-side PL/SQL function, ORDSYS.IM.compatibilityInit( ), at the beginning of the application. See the compatibilityInit( ) method description of this function in this section.

Client-side applications written in Java using Oracle *inter*Media Java Classes for Oracle Database release 8.1.7 or later, should call the OrdMediaUtil.imCompatibilityInit( ) function after connecting to the database.

```
public static void imCompatibilityInit(OracleConnection con)
    throws Exception
```

This Java function takes OracleConnection as an argument. The included *inter*Media release 8.1.7 or later Java API will ensure that your 8.1.7 or later application will work with a potential future release of *inter*Media with evolved object types.

There is not yet a way for client-side PL/SQL applications to maintain compatibility with the current release of the *inter*Media object types if the objects add new attributes in a future release.

See the compatibilityInit( ) method in this section, and *Oracle interMedia Java Classes Reference* for further information, and detailed descriptions and examples. This guide is on the Oracle Technology Network at

http://otn.oracle.com/

compatibilityInit( )

Format

```
compatibilityInit(release IN    VARCHAR2,
                  errmsg OUT VARCHAR2)
RETURN NUMBER;
```

Description

Provides compatibility between software releases by allowing for the evolution of the *inter*Media object types.

Parameters

release
The release number. For example, this string should be set to 10.1 to allow a release 10.1 application to work with a future release of the Oracle Database software and the *inter*Media evolved object types.

errmsg
String output parameter. If the function returns a status other than 0, this errmsg string contains the reason for the failure.

Pragmas

None.

Exceptions

None.

Usage Notes

You should begin using the compatibilityInit( ) method as soon as possible so you will not have to upgrade the Oracle software on your client node, or recompile your client application in order to work with a future release of the Oracle Database software if the *inter*Media object types change in a future release. See Section 2.1 to determine if you need to call this function.

The compatibility initialization function for *inter*Media is located in the ORDSYS.IM package.

Examples

Use OCI and set the compatibilityInit( ) method release parameter to 10.1 to allow a release 10.1 application to work with a future release of the Oracle Database software and evolved *inter*Media object types. This is not a standalone program; it assumes that you have allocated handles beforehand.

```
void prepareExecuteStmt( OCIEnv *envHndl,
             OCIStmt **stmtHndl,
             OCIError *errorHndl,
             OCISvcCtx *serviceCtx,
             OCIBind *bindhp[] )
{
   text *statement = (text *)
      "begin :sts := ORDSYS.IM.compatibilityInit( :vers, :errText );
end;";
  sword sts = 0;
  text *vers = (text *)"10.1";
  text errText[512];
  sb2 nullInd;

  printf( " Preparing statement\n" );

  OCIHandleAlloc( envHndl, (void **) stmtHndl, OCI_HTYPE_STMT, 0, NULL
);

  OCIStmtPrepare( *stmtHndl, errorHndl, (text *)statement,
         (ub4)strlen( (char *)statement ), OCI_NTV_SYNTAX,
         OCI_DEFAULT );

  printf( " Executing statement\n" );

  OCIBindByPos( *stmtHndl, &bindhp[ 0 ], errorHndl, 1, (void *)&sts,
        sizeof( sts ), SQLT_INT, (void *)0, NULL, 0, 0,
        NULL, OCI_DEFAULT );

  OCIBindByPos( *stmtHndl, &bindhp[ 1 ], errorHndl, 2, vers,
        strlen((char *)vers) + 1, SQLT_STR, (void *)0, NULL,
        0, 0, NULL, OCI_DEFAULT );

  OCIBindByPos( *stmtHndl, &bindhp[ 2 ], errorHndl, 3, errText,
        sizeof( errText ), SQLT_STR, &nullInd, NULL, 0, 0,
```

```
        NULL, OCI_DEFAULT);

  OCIStmtExecute( serviceCtx, *stmtHndl, errorHndl, 1, 0,
        (OCISnapshot *)NULL, (OCISnapshot *)NULL, OCI_DEFAULT );

  printf( " Statement executed\n" );
  if (sts != 0)
    {
    printf( "CompatibilityInit failed with Sts = %d\n", sts );
    printf( "%s\n", errText );
    }


}
```

3

Common Methods for *inter*Media Object Types

This chapter presents reference information on the common methods used for the following Oracle *inter*Media ("*inter*Media") object types:

- ORDAudio
- ORDDoc
- ORDImage
- ORDVideo

The examples in this chapter use the ONLINE_MEDIA table in the Product Media sample schema. The examples assume that the table has been populated as shown in examples in Chapter 4, Chapter 6, and Chapter 8.

See *Oracle Database Sample Schemas* for information on the sample schemas.

> **Note:** The *inter*Media methods are designed to be internally consistent. If you use *inter*Media methods (such as import( ) or image process( )) to modify the media data, *inter*Media will ensure that object attributes remain synchronized with the media data. However, if you manipulate the data itself (by either directly modifying the BLOB or changing the external source), then you must ensure that the object attributes stay synchronized and the update time is modified; otherwise, the object attributes will not match the data.

3.1 Embedded ORDSource Object

The ORDSource object is embedded within the ORDAudio, ORDDoc, and ORDVideo object types. The ORDSource object type supports access to a variety of sources of multimedia data. It supports access to data sources locally in a BLOB within Oracle Database, externally from a BFILE on a local file system, externally from a URL on an HTTP server, or externally from a user-defined source on another server. See Chapter 10 for details on how the ORDSource object type is defined, including supported values for the ORDSource attributes, which are the following:

- localData: the locally stored multimedia data stored as a BLOB within the object. Up to 4 gigabytes of data can be stored as a BLOB.
- srcType: the data source type.
- srcLocation: the place where data can be found based on the srcType value.
- srcName: the data object name.
- updateTime: the time at which the data was last updated.
- local: a flag that indicates whether the data is local (1 or NULL) or external (0).

3.2 Important Notes

Methods invoked at the ORDSource level that are handed off to a source plug-in for processing have ctx (RAW) as the first argument. Before calling any of these methods for the first time, the client should allocate the ctx structure, initialize it to NULL, and invoke the openSource( ) method. At this point, the source plug-in can initialize context for this client. When processing is complete, the client should invoke the closeSource( ) method.

Methods invoked at the ORDAudio, ORDDoc, or ORDVideo level that are handed off to a format plug-in for processing have ctx (RAW) as the first argument. Before calling any of these methods for the first time, the client should allocate the ctx structure and initialize it to NULL.

Note: In the current release, none of the plug-ins provided by Oracle and not all source or format plug-ins will use the ctx argument, but if you code as previously described, your application should work with current or future source or format plug-ins.

For ORDAudio, ORDDoc, or ORDVideo object types, you should use any of the individual set methods to set the attribute value for an object for formats not natively supported, or write a format plug-in and call the setProperties( ) method; otherwise, for formats natively supported, use the setProperties( ) method to populate the attributes of the object.

For ORDImage object types, use the setProperties( ) method to populate the attributes of the object. Use the setProperties( ) for foreign images method for formats that are not natively supported.

3.3 Methods

This section presents reference information on the *inter*Media methods that are common to all object types (except Still Image objects).

Other methods, which are particular to a given object type or are implemented differently for each object type, are described in ORDAudio Methods on page 4-13, ORDImage Methods on page 6-12, and ORDVideo Methods on page 8-13.

For more information on object types and methods, see *Oracle Database Concepts*.

The following methods are presented in this section:

clearLocal( )

Format

clearLocal( );

Description

Resets the source.local attribute (of the embedded ORDSource object) to indicate that the data is stored externally. When the source.local attribute is set to 0, media methods look for corresponding data using the source.srcLocation, source.srcName, and source.srcType attributes.

Parameters

None.

Usage Notes

This method sets the source.local attribute to 0, meaning the data is stored externally outside the database.

Pragmas

None.

Exceptions

None.

Examples

Clear the value of the local flag for the data:

```
DECLARE
 obj ORDSYS.ORDAudio;
BEGIN
 SELECT product_audio INTO obj FROM pm.online_media WHERE product_id=1733
  FOR UPDATE;
 obj.clearLocal();
 UPDATE pm.online_media SET product_audio=obj WHERE product_id=1733;
 COMMIT;
 END;
 /
```

closeSource( )

Format

closeSource(ctx IN OUT RAW) RETURN INTEGER;

Description

Closes a data source.

Parameters

ctx
The source plug-in context information. This should be allocated and initialized to NULL. If you are using a user-defined source plug-in, you should call the openSource( ) method. See the introduction to this chapter for more information.

Usage Notes

The RETURN INTEGER is 0 (zero) for success and greater than 0 (for example, 1) for failure. The exact number and the meaning for that number is plug-in defined. For example, for the file plug-in, 1 might mean "File not found," 2 might mean "No such directory," and so forth.

Pragmas

None.

Exceptions

ORDSourceExceptions.INCOMPLETE_SOURCE_INFORMATION

This exception is raised if you call the closeSource( ) method and the value for the source.srcType attribute is NULL.

ORDSourceExceptions.METHOD_NOT_SUPPORTED

This exception is raised if you call the closeSource( ) method and this method is not supported by the source plug-in being used.

ORDSourceExceptions.SOURCE_PLUGIN_EXCEPTION

This exception is raised if you call the closeSource( ) method and the source plug-in raises an exception.

See Appendix F for more information about these exceptions.

Examples

Close an external data source:

```
DECLARE
 obj ORDSYS.ORDAudio;
 res INTEGER;
 ctx RAW(64) :=NULL;
BEGIN
 SELECT product_audio INTO obj FROM pm.online_media WHERE product_id=1733
  FOR UPDATE;
 res := obj.closeSource(ctx);
 UPDATE pm.online_media SET product_audio=obj WHERE product_id=1733;
 COMMIT;
 EXCEPTION
 WHEN ORDSYS.ORDSourceExceptions.METHOD_NOT_SUPPORTED THEN
  DBMS_OUTPUT.PUT_LINE('ORDSourceExceptions.METHOD_NOT_SUPPORTED caught');
 WHEN OTHERS THEN
  DBMS_OUTPUT.PUT_LINE('EXCEPTION caught');
END;
/
```

deleteContent( )

Format

deleteContent( );

Description

Deletes the BLOB from the source.localData attribute (of the embedded ORDSource object), sets the source.local attribute to zero (to indicate that data is not local), and updates the source.updateTime attribute.

Parameters

None.

Usage Notes

This method can be called after you export the data from the local source to an external data source and you no longer need this data in the local source.

Call this method when you want to update the object with a new object.

Pragmas

None.

Exceptions

None.

Examples

Delete the local data from the current local source:

```
DECLARE
 image ORDSYS.ORDImage;
BEGIN
 SELECT product_photo INTO image FROM pm.online_media WHERE product_id = 3515 FOR UPDATE;
 -- Delete the local content of the image:
 Image.deleteContent();
 COMMIT;
END;
/
```

export( )

Format

```
export(
    ctx                IN OUT RAW,
    source_type        IN VARCHAR2,
    source_location    IN VARCHAR2,
    source_name        IN VARCHAR2);
```

Description

Copies data from the BLOB in the source.localData attribute (of the embedded ORDSource object) to a corresponding external data source.

> **Note:** The export( ) method natively supports only sources with a source.srcType value of file. User-defined sources may support the export( ) method.

Parameters

ctx
The source plug-in context information.

source_type
The type of the external source data.

source_location
The location to which the source data is to be exported.

source_name
The name of the object to which the data is to be exported.

Usage Notes

After data is exported, all attributes remain unchanged and source.srcType, source.srcLocation, and source.srcName are updated with input values. After calling the export( ) method, you can call the clearLocal( ) method to indicate the

data is stored outside the database and call the deleteContent( ) method if you want to delete the content of the source.localData attribute.

This method is also available for user-defined sources that can support the export( ) method.

The export( ) method for a source type of file is similar to a file copy operation in that the original data stored in the BLOB is not touched other than for reading purposes.

The export( ) method is not an exact mirror operation to the import( ) method in that the clearLocal( ) method is not automatically called to indicate the data is stored outside the database, whereas the import( ) method automatically calls the setLocal( ) method.

Call the deleteContent( ) method after calling the export( ) method to delete the content from the database if you no longer intend to manage the multimedia data within the database.

The export( ) method writes only to a database directory object that the user has privilege to access. That is, you can access a directory that you have created using the SQL CREATE DIRECTORY statement, or one to which you have been granted READ access. To execute the CREATE DIRECTORY statement, you must have the CREATE ANY DIRECTORY privilege. In addition, you must use the DBMS_JAVA.GRANT_PERMISSION call to specify to which files the user and ORDSYS can write. The user must be granted the write permission so that he or she can write to the file; ORDSYS must be granted the write permission so that it can export the file on behalf of the user. (The installation procedure creates the ORDSYS user by default during installation. See *Oracle interMedia User's Guide* for more information.)

For example, the following SQL*Plus commands grant the user, MEDIAUSER, and ORDSYS the permission to write to the file named filmtrack1.au:

```
CALL DBMS_JAVA.GRANT_PERMISSION(
        'MEDIAUSER',
        'java.io.FilePermission',
        '/audio/movies/filmtrack1.au',
        'write');
CALL DBMS_JAVA.GRANT_PERMISSION(
        'ORDSYS',
        'java.io.FilePermission',
        '/audio/movies/filmtrack1.au',
        'write');
```

The previous example shows how to authorize access to write to a single file. In addition, there are various wildcard path specifications that authorize write access

to multiple directories and file names. For example, a path specification that ends in a slash and asterisk (/*), where the slash is the file-separator character that is operating-system dependent, indicates all the files contained in the specified directory. A path specification that ends with a slash and a hyphen (/-) indicates all files contained in the specified directory and all its subdirectories. A path name consisting of the special token <<ALL FILES>> authorizes access to any file.

See *Oracle Database Java Developer's Guide* and the java.io.FilePermission class in the Java API for more information about security and performance.

Invoking this method implicitly calls the setUpdateTime( ) method.

Pragmas

None.

Exceptions

ORDSourceExceptions.INCOMPLETE_SOURCE_INFORMATION

This exception is raised if you call the export( ) method and the value of the source_type parameter is NULL.

ORDSourceExceptions.METHOD_NOT_SUPPORTED

This exception is raised if you call the export( ) method and this method is not supported by the source plug-in being used.

ORDSourceExceptions.SOURCE_PLUGIN_EXCEPTION

This exception is raised if you call the export( ) method and the source plug-in raises an exception.

See Appendix F for more information about these exceptions.

ORDSourceExceptions.IO_ERROR

This exception is raised if the export( ) method encounters an error writing the BLOB data to the specified operating system file.

Examples

Export data from a local source to an external data source:

```
-- Create the directory to which you want users to export data. Then,
-- grant write access to the directory for ORDSYS and the user who will be
-- doing the exporting, in this case the user is Ron.
connect /as sysdba
```

```
CREATE OR REPLACE DIRECTORY FILE_DIR as '/mydir/work';
GRANT READ ON DIRECTORY FILE_DIR TO PUBLIC WITH GRANT OPTION;
BEGIN
-- Grant permission to the user and ORDSYS.
 DBMS_JAVA.GRANT_PERMISSION(
  'RON',
  'java.io.FilePermission',
  '/images/testimg.jpg',
  'WRITE');
 DBMS_JAVA.GRANT_PERMISSION(
  'ORDSYS',
  'java.io.FilePermission',
  '/images/testimg.jpg',
  'write');
 COMMIT;
END;
/
-- Connect as the user Ron:
CONNECT RON/RON
set serveroutput on;
set echo on;
DECLARE
 obj ORDSYS.ORDImage;
 ctx RAW(64) :=NULL;
BEGIN
SELECT product_photo INTO obj FROM pm.online_media
 WHERE product_id = 3515;
obj.export(ctx,'file','FILE_DIR','testimg.jpg');
COMMIT;
EXCEPTION
 WHEN ORDSYS.ORDSourceExceptions.METHOD_NOT_SUPPORTED THEN
  DBMS_OUTPUT.PUT_LINE('Source METHOD_NOT_SUPPORTED caught');
 WHEN ORDSYS.ORDSourceExceptions.SOURCE_PLUGIN_EXCEPTION THEN
  DBMS_OUTPUT.PUT_LINE('SOURCE PLUGIN EXCEPTION caught');
 WHEN OTHERS THEN
  DBMS_OUTPUT.PUT_LINE('OTHER EXCEPTION caught');
END;
/
```

getBFile( )

Format

getBFile( ) RETURN BFILE;

Description

Returns the LOB locator of the BFILE containing the media.

Parameters

None.

Usage Notes

This method constructs and returns a BFILE using the stored source.srcLocation and source.srcName attribute information (of the embedded ORDSource object). The source.srcLocation attribute must contain a defined directory object. The source.srcName attribute must be a valid file name and source.srcType must be "file".

Pragmas

PRAGMA RESTRICT_REFERENCES(getBFile, WNDS, WNPS, RNDS, RNPS)

Exceptions

ORDSourceExceptions.INCOMPLETE_SOURCE_INFORMATION

This exception is raised if the source.srcType attribute value is NULL.

ORDSourceExceptions.INVALID_SOURCE_TYPE

This exception is raised if the value of the source.srcType attribute is other than "file".

Examples

Return the BFILE for the stored source directory and file name attributes:

```
DECLARE
 obj ORDSYS.ORDVideo;
 videobfile BFILE;
BEGIN
```

```
 SELECT product_video INTO obj FROM pm.online_media
  WHERE product_id = 2030;
 -- Get the video BFILE.
 videobfile := obj.getBFile();
 COMMIT;
 EXCEPTION
  WHEN ORDSYS.ORDSourceExceptions.INCOMPLETE_SOURCE_INFORMATION THEN
   DBMS_OUTPUT.PUT_LINE('The source.srcType attribute value is NULL');
  WHEN ORDSYS.ORDSourceExceptions.INVALID_SOURCE_TYPE THEN
   DBMS_OUTPUT.PUT_LINE('The value of srcType is not file');
END;
/
```

getContent( )

Format

getContent( ) RETURN BLOB;

Description

Returns the BLOB handle to the source.localData attribute (of the embedded ORDSource object).

Parameters

None.

Usage Notes

None.

Pragmas

PRAGMA RESTRICT_REFERENCES(getContent, WNDS, WNPS, RNDS, RNPS)

Exceptions

None.

Examples

Access video data to be put on a Web-based player:

```
DECLARE
 obj ORDSYS.ORDVideo;
 ctx RAW(64) := NULL;
BEGIN
 SELECT product_video INTO obj FROM pm.online_media WHERE product_id = 2030 FOR UPDATE;
 -- import data
 obj.importFrom(ctx,'file','FILE_DIR','printer.rm');
 -- check size
 DBMS_OUTPUT.PUT_LINE('Length is '||TO_CHAR(DBMS_LOB.GETLENGTH(obj.getContent)));
 DBMS_OUTPUT.PUT_LINE(obj.getSource());
 COMMIT;
END;
/
```

getMimeType( )

Format

getMimeType( ) RETURN VARCHAR2;

Description

Returns the MIME type for the data. This is a simple access method that returns the value of the mimeType attribute.

Parameters

None.

Usage Notes

If the source is an HTTP server, the MIME type information is read from the HTTP header information when the media is imported and stored in the object attribute. If the source is a file or BLOB, the MIME type information is extracted when the setProperties( ) method is called.

For unrecognized file formats, users must call the setMimeType( ) method and specify the MIME type.

Use this method rather than accessing the mimeType attribute directly to protect yourself from potential changes to the internal representation of the object.

Pragmas

PRAGMA RESTRICT_REFERENCES(getMimeType, WNDS, WNPS, RNDS, RNPS)

Exceptions

None.

Examples

Get the MIME type for some stored image data:

```
DECLARE
 image ORDSYS.ORDImage;
BEGIN
 SELECT p.product_photo INTO image FROM pm.online_media p
```

```
  WHERE product_id = 3515;
 DBMS_OUTPUT.PUT_LINE('writing mimetype');
 DBMS_OUTPUT.PUT_LINE('----------------');
 DBMS_OUTPUT.PUT_LINE(image.getMimeType);
 COMMIT;
END;
/
```

getSource( )

Format

getSource( ) RETURN VARCHAR2;

Description

Returns information about the external location of the data in URL format. (This information is the source.srcType, source.srcLocation, and source.srcName attribute values of the embedded ORDSource object.)

Parameters

None.

Usage Notes

Possible return values are:

- `FILE://<DIR OBJECT NAME>/<FILE NAME>` for a file source
- `HTTP://<URL>` for an HTTP source
- User-defined source; for example:

  ```
  TYPE://<USER-DEFINED SOURCE LOCATION>/<USER-DEFINED SOURCE
  NAME>
  ```

Pragmas

PRAGMA RESTRICT_REFERENCES(getSource, WNDS, WNPS, RNDS, RNPS)

Exceptions

None.

Examples

Get the source of the image data:

```
DECLARE
 image ORDSYS.ORDImage;
BEGIN
 SELECT p.product_photo INTO image FROM pm.online_media p
```

```
  WHERE p.product_id = 3515;
 -- Get the image source information:
 DBMS_OUTPUT.PUT_LINE(image.getSource);
 COMMIT;
END;
/
```

getSourceLocation( )

Format

getSourceLocation( ) RETURN VARCHAR2;

Description

Returns a string containing the value of the external data source location (the value of the source.srcLocation attribute of the embedded ORDSource object).

Parameters

None.

Usage Notes

This method returns a VARCHAR2 string containing the value of the external data location, for example BFILEDIR.

Pragmas

PRAGMA RESTRICT_REFERENCES(getSourceLocation, WNDS, WNPS, RNDS, RNPS)

Exceptions

ORDSourceExceptions.INCOMPLETE_SOURCE_LOCATION

This exception is raised if you call the getSourceLocation( ) method and the value of the source.srcLocation attribute is NULL.

Examples

Get the source location information about an image data source:

```
DECLARE
 image ORDSYS.ORDImage;
BEGIN
 SELECT p.product_photo INTO image FROM pm.online_media p
  WHERE p.product_id = 3515;
 -- Get the image source location.
 DBMS_OUTPUT.PUT_LINE('Source location is ' || image.getSourceLocation);
 COMMIT;
```

```
END;
/
```

getSourceName( )

Format

getSourceName( ) RETURN VARCHAR2;

Description

Returns a string containing of the name of the external data source (the value of the source.srcName attribute of the embedded ORDSource object).

Parameters

None.

Usage Notes

This method returns a VARCHAR2 string containing the name of the external data source, for example testimg.dat.

Pragmas

PRAGMA RESTRICT_REFERENCES(getSourceName, WNDS, WNPS, RNDS, RNPS)

Exceptions

ORDSourceExceptions.INCOMPLETE_SOURCE_NAME

This exception is raised if you call the getSourceName( ) method and the value of the source.srcName attribute is NULL.

Examples

Get the source name information about an image data source:

```
DECLARE
 image ORDSYS.ORDImage;
BEGIN
 SELECT p.product_photo INTO image FROM pm.online_media p
  WHERE p.product_id = 3515;
 -- Get the image source name.
 DBMS_OUTPUT.PUT_LINE('Source name is ' ||image.getSourceName);
 COMMIT;
```

```
END;
/
```

getSourceType( )

Format

getSourceType( ) RETURN VARCHAR2;

Description

Returns a string containing the type of the external data source (the value of the source.srcType attribute of the embedded ORDSource object).

Parameters

None.

Usage Notes

Returns a VARCHAR2 string containing the type of the external data source, for example "file".

Pragmas

PRAGMA RESTRICT_REFERENCES(getSourceType, WNDS, WNPS, RNDS, RNPS)

Exceptions

None.

Examples

Get the source type information about a media data source:

```
DECLARE
 obj ORDSYS.ORDDoc;
BEGIN
 SELECT p.product_testimonials INTO obj FROM pm.online_media p
  WHERE p.product_id= 3060;
 DBMS_OUTPUT.PUT_LINE('setting and getting source');
 DBMS_OUTPUT.PUT_LINE('-------------------------');
 -- set source to a file
 obj.setSource('file','FILE_DIR','speaker.wav');
 -- get source information
 DBMS_OUTPUT.PUT_LINE('Source is ' || obj.getSource);
 DBMS_OUTPUT.PUT_LINE('Source type is ' || obj.getSourceType);
```

```
 DBMS_OUTPUT.PUT_LINE('Source location is ' || obj.getSourceLocation);
 DBMS_OUTPUT.PUT_LINE('Source name is ' || obj.getSourceName);
 COMMIT;
END;
/
```

getUpdateTime( )

Format

getUpdateTime( ) RETURN DATE;

Description

Returns the time when the object was last updated (the value of the source.updateTime of the embedded ORDSource object).

Parameters

None.

Usage Notes

None.

Pragmas

PRAGMA RESTRICT_REFERENCES(getUpdateTime, WNDS, WNPS, RNDS, RNPS)

Exceptions

None.

Examples

Get the updated time of some audio data:

```
DECLARE
 obj ORDSYS.ORDAudio;
BEGIN
 SELECT p.product_audio INTO obj FROM pm.online_media p
  WHERE p.product_id  = 1733;
 DBMS_OUTPUT.PUT_LINE('Update time is:');
 DBMS_OUTPUT.PUT_LINE(TO_CHAR(obj.getUpdateTime(),'MM-DD-YYYY HH24:MI:SS'));
 COMMIT;
END;
/
```

isLocal( )

Format

isLocal( ) RETURN BOOLEAN;

Description

Returns TRUE if the value of the source.local attribute (of the embedded ORDSource object) is 1, and returns FALSE if the value of the source.local attribute is 0. In other words, returns TRUE if the data is stored in a BLOB in the source.localData attribute or FALSE if the data is stored externally.

Parameters

None.

Usage Notes

None.

Pragmas

PRAGMA RESTRICT_REFERENCES(isLocal, WNDS, WNPS, RNDS, RNPS)

Exceptions

None.

Examples

Determine whether or not the audio data is local:

```
DECLARE
 obj ORDSYS.ORDAudio;
BEGIN
 SELECT p.product_audio INTO obj FROM pm.online_media p
   WHERE p.product_id = 1733;
 IF (obj.isLocal() = TRUE) THEN  DBMS_OUTPUT.PUT_LINE('local is set true');
 ELSE  DBMS_OUTPUT.PUT_LINE('local is set false');
 END IF;
 COMMIT;
END;
/
```

openSource( )

Format

openSource(userArg IN RAW, ctx OUT RAW) RETURN INTEGER;

Description

Opens a data source.

Parameters

userArg
The user argument. This may be used by user-defined source plug-ins.

ctx
The source plug-in context information.

Usage Notes

The return INTEGER is 0 (zero) for success and greater than 0 (for example, 1) for failure. The exact number and the meaning for that number is plug-in defined. For example, for the file plug-in, 1 might mean "File not found," 2 might mean "No such directory," and so forth.

Pragmas

None.

Exceptions

ORDSourceExceptions.INCOMPLETE_SOURCE_INFORMATION

This exception is raised if you call the openSource( ) method and the value for the source.srcType attribute is NULL.

ORDSourceExceptions.METHOD_NOT_SUPPORTED

This exception is raised if you call the openSource( ) method and this method is not supported by the source plug-in being used.

ORDSourceExceptions.SOURCE_PLUGIN_EXCEPTION

This exception is raised if you call the openSource( ) method and the source plug-in raises an exception.

See Appendix F for more information about these exceptions.

Examples

Open an external data source:

```
DECLARE
 obj ORDSYS.ORDAudio;
 res INTEGER;
 ctx RAW(64) :=NULL;
 userArg RAW(64);
BEGIN
 SELECT p.product_audio INTO obj FROM pm.online_media p
  WHERE p.product_id = 1733;
 res := obj.openSource(userArg, ctx);
 COMMIT;
 EXCEPTION
  WHEN ORDSYS.ORDSourceExceptions.METHOD_NOT_SUPPORTED THEN
   DBMS_OUTPUT.PUT_LINE('ORDSourceExceptions.METHOD_NOT_SUPPORTED caught');
  WHEN OTHERS THEN
   DBMS_OUTPUT.PUT_LINE('EXCEPTION caught');
END;
/
```

processSourceCommand( )

Format

```
processSourceCommand(
                        ctx       IN OUT RAW,
                        cmd       IN VARCHAR2,
                        arguments IN VARCHAR2,
                        result    OUT RAW)
RETURN RAW;
```

Description

Lets you send any command and its arguments to the source plug-in. This method is available only for user-defined source plug-ins.

Parameters

ctx
The source plug-in context information. This should be allocated and initialized to NULL. If you are using a user-defined source plug-in, you should call the openSource( ) method. See the introduction to this chapter for more information.

cmd
Any command recognized by the source plug-in.

arguments
The arguments of the command.

result
The result of calling this method returned by the source plug-in.

Usage Notes

Use this method to send any command and its respective arguments to the source plug-in. Commands are not interpreted; they are just taken and passed through to be processed.

Pragmas

None.

Exceptions

ORDSourceExceptions.INCOMPLETE_SOURCE_INFORMATION

This exception is raised if you call the processSourceCommand( ) method and the value of the source.srcType attribute is NULL.

ORDSourceExceptions.METHOD_NOT_SUPPORTED

This exception is raised if you call the processSource( ) method and this method is not supported by the source plug-in being used.

ORDSourceExceptions.SOURCE_PLUGIN_EXCEPTION

This exception is raised if you call the processSourceCommand( ) method and the source plug-in raises an exception.

See Appendix F for more information about these exceptions.

Examples

None.

readFromSource( )

Format

```
readFromSource(
                ctx          IN OUT RAW,
                startPos   IN INTEGER,
                numBytes IN OUT INTEGER,
                buffer       OUT RAW);
```

Description

Lets you read a buffer of *n* bytes from a source beginning at a start position.

Parameters

ctx
The source plug-in context information. This should be allocated and initialized to NULL. If you are using a user-defined source plug-in, you should call the openSource( ) method. See the introduction to this chapter for more information.

startPos
The start position in the data source.

numBytes
The number of bytes to be read from the data source.

buffer
The buffer into which the data will be read.

Usage Notes

This method is not supported for HTTP sources.

To successfully read HTTP source types, you must request that the entire URL source be read. If you want to implement a read method for an HTTP source type, you must provide your own implementation for this method in the modified source plug-in for the HTTP source type.

Pragmas

None.

Exceptions

ORDSourceExceptions.INCOMPLETE_SOURCE_INFORMATION

This exception is raised if you call the readFromSource( ) method and the value of the source.srcType attribute is NULL.

ORDSourceExceptions.NULL_SOURCE

This exception is raised if you call the readFromSource( ) method and the value of source.local is 1 or NULL (TRUE), but the value of the source.localData attribute is NULL.

ORDSourceExceptions.METHOD_NOT_SUPPORTED

This exception is raised if you call the readFromSource( ) method and this method is not supported by the source plug-in being used.

ORDSourceExceptions.SOURCE_PLUGIN_EXCEPTION

This exception is raised if you call the readFromSource( ) method and the source plug-in raises an exception.

See Appendix F for more information about these exceptions.

Examples

Read a buffer from the source:

```
DECLARE
 obj ORDSYS.ORDAudio;
 buffer RAW(4000);
 i INTEGER;
 ctx RAW(64) :=NULL;
BEGIN
 i := 20;
 SELECT p.product_audio into obj from pm.online_media p
  WHERE p.product_id = 1733;
 obj.readFromSource(ctx,1,i,buffer);
 DBMS_OUTPUT.PUT_LINE('Length is ' || TO_CHAR(obj.getContentLength(ctx)));
 COMMIT;
 EXCEPTION
 WHEN ORDSYS.ORDSourceExceptions.METHOD_NOT_SUPPORTED THEN
   DBMS_OUTPUT.PUT_LINE('ORDSourceExceptions.METHOD_NOT_SUPPORTED caught');
```

```
 WHEN ORDSYS.ORDSourceExceptions.INCOMPLETE_SOURCE_INFORMATION THEN
   DBMS_OUTPUT.PUT_LINE('ORDSourceExceptions.INCOMPLETE_SOURCE_INFORMATION
 caught');
 WHEN ORDSYS.ORDSourceExceptions.NULL_SOURCE THEN
   DBMS_OUTPUT.PUT_LINE('ORDSourceExceptions.NULL_SOURCE caught');
 WHEN ORDSYS.ORDSourceExceptions.SOURCE_PLUGIN_EXCEPTION THEN
   DBMS_OUTPUT.PUT_LINE('ORDSourceExceptions.SOURCE_PLUGIN_EXCEPTION caught');
 WHEN OTHERS THEN
   DBMS_OUTPUT.PUT_LINE('EXCEPTION caught');
END;
/
```

setLocal( )

Format

setLocal( );

Description

Sets the source.local attribute (of the embedded ORDSource object) to indicate that the data is stored internally in a BLOB. When the source.local attribute is set, methods look for corresponding data in the source.localData attribute.

Parameters

None.

Usage Notes

This method sets the source.local attribute to 1, meaning the data is stored locally in the source.localData attribute.

Pragmas

None.

Exceptions

<*object-type*>Exceptions.NULL_LOCAL_DATA

This exception is raised if you call the setLocal( ) method and the source.localData attribute value is NULL.

This exception can be raised by ORDAudio, ORDDoc, ORDImage, or ORDVideo object types. Replace <*object-type*> with the object type to which you apply this method.

Examples

Set the flag to local for the data:

```
DECLARE
 obj ORDSYS.ORDAudio;
BEGIN
 SELECT product_audio INTO obj FROM online_media WHERE product_id = 1733;
```

```
 obj.setLocal;
 UPDATE online_media SET product_audio = obj WHERE product_id = 1733;
 COMMIT;
END;
/
```

setMimeType( )

Format

setMimeType(mime IN VARCHAR2);

Description

Lets you set the MIME type of the data.

Parameters

mime
The MIME type.

Usage Notes

You can override the automatic setting of MIME information by calling this method with a specified MIME value.

Calling this method implicitly calls the setUpdateTime( ) method.

The method setProperties( ) calls this method implicitly.

For image objects, the methods process( ) and processCopy( ) also call this method implicitly.

Pragmas

None.

Exceptions

<*object-type*>Exceptions.INVALID_MIME_TYPE

This exception is raised if you call the setMimeType( ) method and the value for the mime parameter is NULL.

This exception can be raised by ORDAudio, ORDDoc, or ORDVideo object types. Replace <*object-type*> with the object type to which you apply this method.

Examples

Set the MIME type for some stored data:

```
DECLARE
  obj ORDSYS.ORDAudio;
BEGIN
 SELECT p.product_audio INTO obj FROM pm.online_media p
  WHERE p.product_id = 1733 FOR UPDATE;
 DBMS_OUTPUT.PUT_LINE('writing current mimetype');
 DBMS_OUTPUT.PUT_LINE('----------------');
 DBMS_OUTPUT.PUT_LINE(obj.getMimeType);
 DBMS_OUTPUT.PUT_LINE('setting and writing new mimetype');
 DBMS_OUTPUT.PUT_LINE('----------------');
 obj.setMimeType('audio/basic');
 DBMS_OUTPUT.PUT_LINE(obj.getMimeType);
 UPDATE pm.online_media p SET p.product_audio = obj WHERE p.product_id = 1733;
 COMMIT;
END;
/
```

setSource( )

Format

```
setSource(
        source_type      IN VARCHAR2,
        source_location IN VARCHAR2,
        source_name      IN VARCHAR2);
```

Description

Sets or alters information about the external source of the data.

Parameters

source_type
The type of the external source data. See the ORDSource Object Type definition in Chapter 10 for more information.

source_location
The location of the external source data. See the ORDSource Object Type definition in Chapter 10 for more information.

source_name
The name of the external source data. See the ORDSource Object Type definition in Chapter 10 for more information.

Usage Notes

Users can use this method to set the data source to a new FILE or URL.

You must ensure that the directory indicated by the source_location parameter exists or is created before you use this method.

Calling this method implicitly calls the source.setUpdateTime( ) method and the clearLocal( ) method.

Pragmas

None.

Exceptions

ORDSourceExceptions.INCOMPLETE_SOURCE_INFORMATION

This exception is raised if you call the setSource( ) method and the value of the source.srcType attribute is NULL.

Examples

Set the source of the data:

```
DECLARE
 obj ORDSYS.ORDAudio;
BEGIN
 SELECT p.product_audio INTO obj FROM pm.online_media p
  WHERE p.product_id = 1733 FOR UPDATE;
 DBMS_OUTPUT.PUT_LINE('setting and getting source');
 DBMS_OUTPUT.PUT_LINE('-------------------------');
 obj.setSource('file','FILE_DIR','audio.au');
 DBMS_OUTPUT.PUT_LINE(obj.getSource());
 UPDATE pm.online_media p SET p.product_audio = obj WHERE p.product_id = 1733;
 COMMIT;
END;
/
```

setUpdateTime( )

Format

setUpdateTime(current_time DATE);

Description

Sets the time when the data was last updated (the source.srcUpdateTime attribute of the embedded ORDSource object). Use this method whenever you modify the data. Methods that modify the object attributes and all set media access methods call this method implicitly. For example, the methods setMimeType( ), setSource( ), and deleteContent( ) call this method explicitly.

Parameters

current_time
The timestamp to be stored. Defaults to SYSDATE.

Usage Notes

You must invoke this method whenever you modify the data without using object methods.

Pragmas

None.

Exceptions

None.

Examples

Set the updated time of some data:

```
DECLARE
 obj ORDSYS.ORDAudio;
BEGIN
SELECT p.product_audio INTO obj FROM pm.online_media p
 WHERE p.product_id = 1733 FOR UPDATE;
DBMS_OUTPUT.PUT_LINE('current update time:');
DBMS_OUTPUT.PUT_LINE(obj.getUpdateTime);
```

```
DBMS_OUTPUT.PUT_LINE('set and get new update time:');
 obj.setUpdateTime(SYSDATE);
 DBMS_OUTPUT.PUT_LINE(obj.getUpdateTime);
 UPDATE pm.online_media p SET p.product_audio = obj WHERE p.product_id = 1733;
 COMMIT;
END;
/
```

trimSource( )

Format

```
trim(ctx       IN OUT RAW,
     newlen IN INTEGER) RETURN INTEGER;
```

Description

Trims a data source.

Parameters

ctx
The source plug-in context information. This should be allocated and initialized to NULL. If you are using a user-defined source plug-in, you should call the openSource( ) method. See the introduction to this chapter for more information.

newlen
The trimmed new length.

Usage Notes

The return INTEGER is 0 (zero) for success and greater than 0 (for example, 1) for failure. The exact number and the meaning for that number is plug-in defined. For example, for the file plug-in, 1 might mean "File not found," 2 might mean "No such directory," and so forth.

Pragmas

None.

Exceptions

ORDSourceExceptions.INCOMPLETE_SOURCE_INFORMATION

This exception is raised if you call the trimSource( ) method and the value for the source.srcType attribute is NULL.

ORDSourceExceptions.METHOD_NOT_SUPPORTED

This exception is raised if you call the trimSource( ) method and this method is not supported by the source plug-in being used.

ORDSourceExceptions.SOURCE_PLUGIN_EXCEPTION

This exception is raised if you call the trimSource( ) method and the source plug-in raises an exception.

See Appendix F for more information about these exceptions.

Examples

Trim an external data source:

```
DECLARE
 obj ORDSYS.ORDAudio;
 res INTEGER;
 ctx RAW(64) :=NULL;
BEGIN
 SELECT p.product_audio INTO obj FROM pm.online_media p
  WHERE p.product_id = 1733 FOR UPDATE;
 res := obj.trimSource(ctx,0);
 UPDATE pm.online_media p SET p.product_audio = obj WHERE p.product_id = 1733;
 COMMIT;
 EXCEPTION
  WHEN ORDSYS.ORDSourceExceptions.METHOD_NOT_SUPPORTED THEN
   DBMS_OUTPUT.PUT_LINE('ORDSourceExceptions.METHOD_NOT_SUPPORTED caught');
 WHEN ORDSYS.ORDSourceExceptions.INCOMPLETE_SOURCE_INFORMATION THEN
   DBMS_OUTPUT.PUT_LINE('ORDSourceExceptions.INCOMPLETE_SOURCE_INFORMATION
 caught');
 WHEN ORDSYS.ORDSourceExceptions.SOURCE_PLUGIN_EXCEPTION THEN
   DBMS_OUTPUT.PUT_LINE('ORDSourceExceptions.SOURCE_PLUGIN_EXCEPTION caught');
 WHEN OTHERS THEN
   DBMS_OUTPUT.PUT_LINE('EXCEPTION caught');
END;
/
```

writeToSource( )

Format

```
writeToSource(
                ctx          IN OUT RAW,
                startPos     IN INTEGER,
                numBytes IN OUT INTEGER,
                buffer       IN RAW);
```

Description

Lets you write a buffer of *n* bytes to a source beginning at a start position.

Parameters

ctx
The source plug-in context information. This should be allocated and initialized to NULL. If you are using a user-defined source plug-in, you should call the openSource( ) method. See the introduction to this chapter for more information.

startPos
The start position in the source to where the buffer should be copied.

numBytes
The number of bytes to be written to the source.

buffer
The buffer of data to be written.

Usage Notes

This method assumes that the source lets you write *n* number of bytes starting at a random byte location. The file and HTTP source types will not permit you to write, and do not support this method. This method will work if data is stored in a local BLOB or is accessible through a user-defined source plug-in.

Pragmas

None.

Exceptions

ORDSourceExceptions.INCOMPLETE_SOURCE_INFORMATION

This exception is raised if you call the writeToSource( ) method and the value of the source.srcType attribute is NULL.

<*object-type*>Exceptions.NULL_SOURCE

This exception is raised if you call the writeToSource( ) method and the value of source.local is 1 or NULL (TRUE), but the value of the source.localData attribute is NULL.

Replace <object-type> with the object type to which you apply this method.

ORDSourceExceptions.METHOD_NOT_SUPPORTED

This exception is raised if you call the writeToSource( ) method and this method is not supported by the source plug-in being used.

ORDSourceExceptions.SOURCE_PLUGIN_EXCEPTION

This exception is raised if you call the writeToSource( ) method and the source plug-in raises an exception.

See Appendix F for more information about these exceptions.

Examples

Write a buffer to the source:

```
DECLARE
 obj ORDSYS.ORDAudio;
 n INTEGER := 6;
 ctx RAW(64) :=NULL;
BEGIN
 SELECT p.product_audio INTO obj FROM pm.online_media p
  WHERE p.product_id = 1743 FOR UPDATE;
 obj.writeToSource(ctx,1,n,UTL_RAW.CAST_TO_RAW('helloP'));
 UPDATE pm.online_media p SET p.product_audio = obj WHERE p.product_id = 1743;
 DBMS_OUTPUT.PUT_LINE('Length is ' || TO_CHAR(obj.getContentLength(ctx)));
 -- Roll back the transaction to keep the sample schema unchanged.
 ROLLBACK;
 EXCEPTION
  WHEN ORDSYS.ORDSourceExceptions.METHOD_NOT_SUPPORTED THEN
```

```
   DBMS_OUTPUT.PUT_LINE('ORDSourceExceptions.METHOD_NOT_SUPPORTED caught');
  WHEN OTHERS THEN DBMS_OUTPUT.PUT_LINE('EXCEPTION caught');
END;
/
```

4

ORDAudio

Oracle *inter*Media *("inter*Media") contains information about the ORDAudio object type:

The examples in this chapter use the ONLINE_MEDIA table in the Product Media sample schema. To replicate the examples on your own computer, you should begin with the examples shown in the reference pages for the ORDAudio constructors and the import( ) and importFrom( ) methods. Substitute audio files you have for the ones shown in the examples. In addition, for a user "ron" to use the examples, the following statements must be issued before ron executes the examples, where "/mydir/work" is the directory where ron will find the audio data:

```
CONNECT /as sysdba
CREATE OR REPLACE DIRECTORY FILE_DIR as '/mydir/work';
GRANT READ ON DIRECTORY FILE_DIR TO PUBLIC WITH GRANT OPTION;
```

See *Oracle Database Sample Schemas* for information on the sample schemas.

> **Note:** If you manipulate the audio data itself (by either directly modifying the BLOB or changing the external source), then you must ensure that the object attributes stay synchronized and the update time is modified; otherwise, the object attributes will not match the audio data.

Methods invoked at the ORDSource level that are handed off to the source plug-in for processing have ctx (RAW) as the first argument. Before calling any of these

methods for the first time, the client should allocate the ctx structure, initialize it to NULL, and invoke the openSource( ) method. At this point, the source plug-in can initialize context for this client. When processing is complete, the client should invoke the closeSource( ) method.

Methods invoked from a source plug-in call have the first argument as ctx (RAW).

Methods invoked at the ORDAudio level that are handed off to the format plug-in for processing have ctx (RAW) as the first argument. Before calling any of these methods for the first time, the client should allocate the ctx structure and initialize it to NULL.

Note: In the current release, none of the plug-ins provided by Oracle and not all source or format plug-ins will use the ctx argument, but if you code as previously described, your application should work with current or future source or format plug-ins.

You should use any of the individual set methods to set the attribute value for an object for formats not natively supported; otherwise, for formats natively supported, use the setProperties( ) method to populate the attributes of the object or write a format plug-in.

ORDAudio Object Type

The ORDAudio object type supports the storage and management of audio data. This object type is defined as follows:

```
CREATE OR REPLACE TYPE ORDAudio
AS OBJECT
(
  -- ATTRIBUTES
description         VARCHAR2(4000),
source              ORDSource,
format              VARCHAR2(31),
mimeType            VARCHAR2(4000),
comments            CLOB,
  -- AUDIO RELATED ATTRIBUTES
encoding            VARCHAR2(256),
numberOfChannels    INTEGER,
samplingRate        INTEGER,
sampleSize          INTEGER,
compressionType     VARCHAR2(4000),
audioDuration       INTEGER,

  -- METHODS
-- CONSTRUCTORS
--
STATIC FUNCTION init( ) RETURN ORDAudio,
STATIC FUNCTION init(srcType     IN VARCHAR2,
                     srcLocation IN VARCHAR2,
                     srcName     IN VARCHAR2) RETURN ORDAudio,
-- Methods associated with the date attribute
MEMBER FUNCTION getUpdateTime( ) RETURN DATE,
PRAGMA RESTRICT_REFERENCES(getUpdateTime, WNDS, WNPS, RNDS, RNPS),
MEMBER PROCEDURE setUpdateTime(current_time DATE),
-- Methods associated with the description attribute
MEMBER PROCEDURE setDescription(user_description IN VARCHAR2),
MEMBER FUNCTION getDescription( ) RETURN VARCHAR2,
PRAGMA RESTRICT_REFERENCES(getDescription, WNDS, WNPS, RNDS, RNPS),

-- Methods associated with the mimeType attribute
MEMBER PROCEDURE setMimeType(mime IN VARCHAR2),
MEMBER FUNCTION getMimeType( ) RETURN VARCHAR2,
PRAGMA RESTRICT_REFERENCES(getMimeType, WNDS, WNPS, RNDS, RNPS),
```

```
-- Methods associated with the source attribute
MEMBER FUNCTION processSourceCommand(
                                    ctx       IN OUT RAW,
                                    cmd       IN VARCHAR2,
                                    arguments IN VARCHAR2,
                                    result    OUT RAW)
               RETURN RAW,

MEMBER FUNCTION  isLocal( ) RETURN BOOLEAN,
PRAGMA RESTRICT_REFERENCES(isLocal, WNDS, WNPS, RNDS, RNPS),

MEMBER PROCEDURE  setLocal( ),
MEMBER PROCEDURE  clearLocal( ),

MEMBER PROCEDURE setSource(
                          source_type     IN VARCHAR2,
                          source_location IN VARCHAR2,
                          source_name     IN VARCHAR2),
MEMBER FUNCTION getSource( ) RETURN VARCHAR2,
PRAGMA RESTRICT_REFERENCES(getSource, WNDS, WNPS, RNDS, RNPS),

MEMBER FUNCTION getSourceType( ) RETURN VARCHAR2,
PRAGMA RESTRICT_REFERENCES(getSourceType, WNDS, WNPS, RNDS, RNPS),

MEMBER FUNCTION getSourceLocation( ) RETURN VARCHAR2,
PRAGMA RESTRICT_REFERENCES(getSourceLocation, WNDS, WNPS, RNDS, RNPS),

MEMBER FUNCTION getSourceName( ) RETURN VARCHAR2,
PRAGMA RESTRICT_REFERENCES(getSourceName, WNDS, WNPS, RNDS, RNPS),

MEMBER PROCEDURE import(ctx IN OUT RAW),
MEMBER PROCEDURE importFrom(
                           ctx             IN OUT RAW,
                           source_type     IN VARCHAR2,
                           source_location IN VARCHAR2,
                           source_name     IN VARCHAR2),
MEMBER PROCEDURE export(
                       ctx             IN OUT RAW,
                       source_type     IN VARCHAR2,
                       source_location IN VARCHAR2,
                       source_name     IN VARCHAR2),
MEMBER FUNCTION getContentLength(ctx IN OUT RAW) RETURN INTEGER,
PRAGMA RESTRICT_REFERENCES(getContentLength, WNDS, WNPS, RNDS, RNPS),

MEMBER PROCEDURE getContentInLob(
```

```
                                ctx      IN OUT RAW,
                                dest_lob IN OUT NOCOPY BLOB,
                                mimeType OUT VARCHAR2,
                                format   OUT VARCHAR2),

MEMBER FUNCTION getContent( ) RETURN BLOB,
PRAGMA RESTRICT_REFERENCES(getContent, WNDS, WNPS, RNDS, RNPS),

MEMBER PROCEDURE deleteContent( ),

MEMBER FUNCTION getBFILE( ) RETURN BFILE,
PRAGMA RESTRICT_REFERENCES(getBFILE, WNDS, WNPS, RNDS, RNPS),

-- Methods associated with file operations on the source
MEMBER FUNCTION openSource(userArg IN RAW, ctx OUT RAW) RETURN INTEGER,
MEMBER FUNCTION closeSource(ctx IN OUT RAW) RETURN INTEGER,
MEMBER FUNCTION trimSource(ctx     IN OUT RAW,
                           newlen  IN INTEGER) RETURN INTEGER,
MEMBER PROCEDURE readFromSource(
                                ctx      IN OUT RAW,
                                startPos IN INTEGER,
                                numBytes IN OUT INTEGER,
                                buffer   OUT RAW),
MEMBER PROCEDURE writeToSource(
                               ctx      IN OUT RAW,
                               startPos IN INTEGER,
                               numBytes IN OUT INTEGER,
                               buffer   IN RAW),

-- Methods associated with audio attributes accessors
MEMBER PROCEDURE setFormat(knownformat IN VARCHAR2),
MEMBER FUNCTION getFormat( ) RETURN VARCHAR2,
PRAGMA RESTRICT_REFERENCES(getFormat, WNDS, WNPS, RNDS, RNPS),

MEMBER PROCEDURE setEncoding(knownEncoding IN VARCHAR2),
MEMBER FUNCTION getEncoding( ) RETURN VARCHAR2,
PRAGMA RESTRICT_REFERENCES(getEncoding, WNDS, WNPS, RNDS, RNPS),

MEMBER PROCEDURE setNumberOfChannels(knownNumberOfChannels IN INTEGER),
MEMBER FUNCTION getNumberOfChannels( ) RETURN INTEGER,
PRAGMA RESTRICT_REFERENCES(getNumberOfChannels, WNDS, WNPS, RNDS, RNPS),

MEMBER PROCEDURE setSamplingRate(knownSamplingRate IN INTEGER),
MEMBER FUNCTION getSamplingRate( ) RETURN INTEGER,
PRAGMA RESTRICT_REFERENCES(getSamplingRate, WNDS, WNPS, RNDS, RNPS),
```

```
MEMBER PROCEDURE setSampleSize(knownSampleSize IN INTEGER),
MEMBER FUNCTION getSampleSize( ) RETURN INTEGER,
PRAGMA RESTRICT_REFERENCES(getSampleSize, WNDS, WNPS, RNDS, RNPS),

MEMBER PROCEDURE setCompressionType(knownCompressionType IN VARCHAR2),
MEMBER FUNCTION getCompressionType( ) RETURN VARCHAR2,
PRAGMA RESTRICT_REFERENCES(getCompressionType, WNDS, WNPS, RNDS, RNPS),

MEMBER PROCEDURE setAudioDuration(knownAudioDuration IN INTEGER),
MEMBER FUNCTION getAudioDuration( ) RETURN INTEGER,
PRAGMA RESTRICT_REFERENCES(getAudioDuration, WNDS, WNPS, RNDS, RNPS),

MEMBER PROCEDURE setKnownAttributes(
                                  knownFormat IN VARCHAR2,
                                  knownEncoding IN VARCHAR2,
                                  knownNumberOfChannels IN INTEGER,
                                  knownSamplingRate IN INTEGER,
                                  knownSampleSize IN INTEGER,
                                  knownCompressionType IN VARCHAR2,
                                  knownAudioDuration IN INTEGER),

-- Methods associated with setting all the properties
MEMBER PROCEDURE setProperties(ctx         IN OUT RAW,
                               setComments IN BOOLEAN),
MEMBER FUNCTION checkProperties(ctx IN OUT RAW) RETURN BOOLEAN,

MEMBER FUNCTION getAttribute(
                            ctx  IN OUT RAW,
                            name IN VARCHAR2) RETURN VARCHAR2,

MEMBER PROCEDURE getAllAttributes(
                                 ctx        IN OUT RAW,
                                 attributes IN OUT NOCOPY CLOB),

-- Methods associated with audio processing
MEMBER FUNCTION processAudioCommand(
                                   ctx       IN OUT RAW,
                                   cmd       IN VARCHAR2,
                                   arguments IN VARCHAR2,
                                   result    OUT RAW)
              RETURN RAW
);
```

where:

- description: the description of the audio object.
- source: the ORDSource where the audio data is to be found.
- format: the format in which the audio data is stored.
- mimeType: the MIME type information.
- comments: the metadata information of the audio object.
- encoding: the encoding type of the audio data.
- numberOfChannels: the number of audio channels in the audio data.
- samplingRate: the rate in Hz at which the audio data was recorded.
- sampleSize: the sample width or number of samples of audio in the data.
- compressionType: the compression type of the audio data.
- audioDuration: the total duration of the audio data stored.

Note: The comments attribute is populated by the setProperties( ) method when the setComments parameter is TRUE and by Oracle *inter*Media Annotator. Oracle Corporation recommends that you not write to this attribute directly.

ORDAudio Constructors

This section describes the ORDAudio constructor functions, which are the following:

init( ) for ORDAudio

Format

init( ) RETURN ORDAudio;

Description

Initializes instances of the ORDAudio object type.

Parameters

None.

Pragmas

None.

Exceptions

None.

Usage Notes

This constructor is a static method that initializes all the ORDAudio attributes to NULL with the following exceptions:

- source.updateTime is set to SYSDATE
- source.local is set to 1 (local)
- source.localData is set to empty_blob

You should begin using the init( ) method as soon as possible to allow you to more easily initialize the ORDAudio object type, especially if the ORDAudio type evolves and attributes are added in a future release. INSERT statements left unchanged using the default constructor (which initializes each object attribute), will fail under these circumstances.

Examples

Initialize the ORDAudio object attributes:

```
BEGIN
 INSERT INTO pm.online_media (product_id, product_audio)
 VALUES (1729, ORDSYS.ORDAudio.init());
```

```
 COMMIT;
END;
/
```

init(srcType,srcLocation,srcName) for ORDAudio

Format

```
init(srcType     IN VARCHAR2,
   srcLocation IN VARCHAR2,
   srcName     IN VARCHAR2)
 RETURN ORDAudio;
```

Description

Initializes instances of the ORDAudio object type.

Parameters

srcType
The source type of the audio data.

srcLocation
The source location of the audio data.

srcName
The source name of the audio data.

Pragmas

None.

Exceptions

None.

Usage Notes

This constructor is a static method that initializes all the ORDAudio attributes to NULL with the following exceptions:

- source.updateTime is set to SYSDATE
- source.local is set to 0
- source.localData is set to empty_blob

- source.srcType is set to the input value
- source.srcLocation is set to the input value
- source.srcName is set to the input value

You should begin using the init( ) method as soon as possible to allow you to more easily initialize the ORDAudio object type, especially if the ORDAudio type evolves and attributes are added in a future release. INSERT statements left unchanged using the default constructor (which initializes each object attribute), will fail under these circumstances.

Examples

Initialize the ORDAudio object attributes:

```
BEGIN
   INSERT INTO pm.online_media (product_id, product_audio)
  VALUES (1733, ORDSYS.ORDAudio.init('FILE', 'FILE_DIR','speaker.au'));
 COMMIT;
END;
/
```

ORDAudio Methods

This section presents reference information on the Oracle *inter*Media methods used specifically for audio data manipulation.

Chapter 3 presents reference information on the Oracle *inter*Media methods that are common to ORDAudio, ORDDoc, ORDImage, and ORDVideo. Use the methods presented in both chapters to get and set attributes, and to perform metadata extractions.

For more information on object types and methods, see *Oracle Database Concepts*.

The following methods are presented in this section:

checkProperties( )

Format

checkProperties(ctx IN OUT RAW) RETURN BOOLEAN;

Description

Checks the properties of the stored audio data, including the following audio attributes: sample size, sample rate, number of channels, format, and encoding type.

Parameters

ctx
The format plug-in context information.

Usage Notes

If the value of the format is set to NULL, then the checkProperties( ) method uses the default format plug-in; otherwise, it uses the plug-in specified by the format.

The checkProperties( ) method does not check the MIME type because a file can have multiple correct MIME types.

Pragmas

None.

Exceptions

ORDAudioExceptions.AUDIO_PLUGIN_EXCEPTION

This exception is raised if you call the checkProperties( ) method and the audio plug-in raises an exception.

Examples

Check property information for known audio attributes:

```
DECLARE
 obj ORDSYS.ORDAudio;
 ctx RAW(64) :=NULL;
BEGIN
 SELECT p.product_audio INTO obj FROM pm.online_media p
 WHERE p.product_id  = 1729;
```

```
 IF ( obj.checkProperties(ctx) =  TRUE ) THEN
  DBMS_OUTPUT.PUT_LINE('true');
 ELSE
  DBMS_OUTPUT.PUT_LINE('false');
 END IF;
 COMMIT;
 EXCEPTION
  WHEN ORDSYS.ORDAudioExceptions.AUDIO_PLUGIN_EXCEPTION THEN
   DBMS_OUTPUT.PUT_LINE('ORDAudioExceptions.AUDIO_PLUGIN_EXCEPTION caught');
  WHEN OTHERS THEN
   DBMS_OUTPUT.PUT_LINE('EXCEPTION caught');
END;
/
```

getAllAttributes( )

Format

```
getAllAttributes(
                ctx          IN OUT RAW,
                attributes IN OUT NOCOPY CLOB);
```

Description

Returns a formatted string for convenient client access. For natively supported formats, the string includes the following list of audio data attributes separated by a comma (,): fileFormat, mimeType, encoding, numberOfChannels, samplingRate, sampleSize, compressionType, and audioDuration. For user-defined formats, the string is defined by the format plug-in.

Parameters

ctx
The format plug-in context information.

attributes
The attributes.

Usage Notes

Generally, these audio data attributes are available from the header of the formatted audio data.

Pragmas

None.

Exceptions

ORDAudioExceptions.AUDIO_PLUGIN_EXCEPTION

This exception is raised if you call the getAllAttributes( ) method and the audio plug-in raises an exception.

Examples

Return all audio attributes for audio data stored in the database:

```
DECLARE
 obj ORDSYS.ORDAudio;
 tempLob   CLOB;
 ctx RAW(64) :=NULL;
BEGIN
 SELECT p.product_audio INTO obj FROM pm.online_media p
  WHERE p.product_id  = 1729;
 DBMS_OUTPUT.PUT_LINE('getting comma separated list of all attributes');
 DBMS_OUTPUT.PUT_LINE('----------------------------------------');
 DBMS_LOB.CREATETEMPORARY(tempLob, FALSE, DBMS_LOB.CALL);
 obj.getAllAttributes(ctx,tempLob);
 DBMS_OUTPUT.PUT_LINE(DBMS_LOB.substr(tempLob, DBMS_LOB.getLength(tempLob),1));
 COMMIT;
 EXCEPTION
  WHEN ORDSYS.ORDAudioExceptions.AUDIO_PLUGIN_EXCEPTION THEN
   DBMS_OUTPUT.PUT_LINE('ORDAudioExceptions.AUDIO_PLUGIN_EXCEPTION caught');
END;
/
```

getAttribute( )

Format

```
getAttribute(
        ctx     IN OUT RAW,
        name IN VARCHAR2)
RETURN VARCHAR2;
```

Description

Returns the value of the requested attribute from audio data for user-defined formats only.

Parameters

ctx
The format plug-in context information.

name
The name of the attribute.

Usage Notes

Generally, the audio data attributes are available from the header of the formatted audio data.

Audio data attribute information can be extracted from the audio data itself. You can extend support to a format not understood by the ORDAudio object by implementing an ORDPLUGINS.ORDX_<format>_AUDIO package that supports that format. See *Oracle interMedia User's Guide* for more information.

Pragmas

None.

Exceptions

ORDAudioExceptions.AUDIO_PLUGIN_EXCEPTION

This exception is raised if you call the getAttribute( ) method and the audio plug-in raises an exception.

Examples

Return information for the specified audio attribute for audio data stored in the database. (Because this example uses a supported data format, rather than a user-written plug-in, an exception will be raised.)

```
DECLARE
 obj ORDSYS.ORDAudio;
 res VARCHAR2(4000);
 ctx RAW(64) :=NULL;
BEGIN
 SELECT p.product_audio INTO obj FROM pm.online_media p
  WHERE p.product_id  = 1733;
 DBMS_OUTPUT.PUT_LINE('getting audio sample size');
 DBMS_OUTPUT.PUT_LINE('----------------------');
 res := obj.getAttribute(ctx,'sample_size');
 COMMIT;
 EXCEPTION
  WHEN ORDSYS.ORDAudioExceptions.AUDIO_PLUGIN_EXCEPTION THEN
   DBMS_OUTPUT.PUT_LINE('AUDIO PLUGIN EXCEPTION caught');
  WHEN OTHERS THEN
   DBMS_OUTPUT.PUT_LINE('EXCEPTION caught');
END;
/
```

getAudioDuration( )

Format

```
getAudioDuration( ) RETURN INTEGER;
```

Description

Returns the value of the audioDuration attribute of the audio object.

Parameters

None.

Usage Notes

None.

Pragmas

PRAGMA RESTRICT_REFERENCES(getAudioDuration, WNDS, WNPS, RNDS, RNPS)

Exceptions

None.

Examples

See the example in setKnownAttributes( ) on page 4-47.

getCompressionType( )

Format

getCompressionType( ) RETURN VARCHAR2;

Description

Returns the value of the compressionType attribute of the audio object.

Parameters

None.

Usage Notes

None.

Pragmas

PRAGMA RESTRICT_REFERENCES(getCompressionType, WNDS, WNPS, RNDS, RNPS)

Exceptions

None.

Examples

See the example in setKnownAttributes( ) on page 4-48.

getContentLength( )

Format

```
getContentLength(ctx IN OUT RAW) RETURN INTEGER;
```

Description

Returns the length of the audio data content stored in the source.

Parameters

ctx
The source plug-in context information.

Usage Notes

None.

Pragmas

PRAGMA RESTRICT_REFERENCES(getContentLength, WNDS, WNPS, RNDS, RNPS)

Exceptions

ORDSourceExceptions.INCOMPLETE_SOURCE_INFORMATION

This exception is raised if you call the getContentLength( ) method and the value of the source.srcType attribute is NULL.

ORDSourceExceptions.SOURCE_PLUGIN_EXCEPTION

This exception is raised if you call the getContentLength( ) method and the source plug-in raises an exception.

See Appendix F for more information about these exceptions.

Examples

See the example in import( ) on page 4-33.

getContentInLob( )

Format

```
getContentInLob(
                ctx          IN OUT RAW,
                dest_lob     IN OUT NOCOPY BLOB,
                mimeType     OUT VARCHAR2,
                format       OUT VARCHAR2);
```

Description

Copies data from a data source into the specified BLOB. The BLOB must not be the BLOB in the source.localData attribute (of the embedded ORDSource object).

Parameters

ctx
The source plug-in context information.

dest_lob
The LOB in which to receive data.

mimeType
The MIME type of the data; this may or may not be returned.

format
The format of the data; this may or may not be returned.

Usage Notes

None.

Pragmas

None.

Exceptions

ORDSourceExceptions.METHOD_NOT_SUPPORTED

This exception is raised if you call the getContentInLob( ) method and this method is not supported by the source plug-in being used.

ORDSourceExceptions.SOURCE_PLUGIN_EXCEPTION

This exception is raised if you call the getContentInLob( ) method and the source plug-in raises an exception.

See Appendix F for more information about these exceptions.

Examples

Get data from a data source and put it into the specified BLOB:

```
DECLARE
 obj ORDSYS.ORDAudio;
 tempBLob BLOB;
 mimeType VARCHAR2(4000);
 format VARCHAR2(31);
 ctx RAW(64) :=NULL;
BEGIN
 SELECT p.product_audio INTO obj FROM pm.online_media p
  WHERE p.product_id  = 1733;
 IF (obj.isLocal) THEN
  DBMS_OUTPUT.PUT_LINE('local is true');
 END IF;
 DBMS_LOB.CREATETEMPORARY(tempBLob, true, 10);
 obj.getContentInLob(ctx,tempBLob, mimeType,format);
 DBMS_OUTPUT.PUT_LINE('Length is ' || TO_CHAR(DBMS_LOB.getLength(tempBLob)));
 COMMIT;
 EXCEPTION
  WHEN ORDSYS.ORDSourceExceptions.METHOD_NOT_SUPPORTED THEN
   DBMS_OUTPUT.PUT_LINE('ORDSourceExceptions.METHOD_NOT_SUPPORTED caught');
  WHEN OTHERS THEN
   DBMS_OUTPUT.PUT_LINE('EXCEPTION caught');
 END;
/
```

getDescription( )

Format

getDescription( ) RETURN VARCHAR2;

Description

Returns the description of the audio data.

Parameters

None.

Usage Notes

None.

Pragmas

PRAGMA RESTRICT_REFERENCES(getDescription, WNDS, WNPS, RNDS, RNPS)

Exceptions

ORDAudioExceptions.DESCRIPTION_IS_NOT_SET

This exception is raised if you call the getDescription( ) method and the description is not set.

Examples

Get the description attribute for some audio data:

```
DECLARE
  obj ORDSYS.ORDAudio;
BEGIN
-- This example assumes that the setDescription method has already been applied.
  SELECT p.product_audio INTO obj FROM pm.online_media p
    WHERE p.product_id  = 1733 FOR UPDATE;
  DBMS_OUTPUT.PUT_LINE('Current description is:');
  DBMS_OUTPUT.PUT_LINE('-------------');
  DBMS_OUTPUT.PUT_LINE(obj.getDescription());
  COMMIT;
END;
/
```

getEncoding( )

Format

getEncoding( ) RETURN VARCHAR2;

Description

Returns the value of the encoding attribute of the audio object.

Parameters

None.

Usage Notes

None.

Pragmas

PRAGMA RESTRICT_REFERENCES(getEncoding, WNDS, WNPS, RNDS, RNPS)

Exceptions

None.

Examples

See the example in setProperties( ) on page 4-51.

getFormat( )

Format

getFormat( ) RETURN VARCHAR2;

Description

Returns the value of the format attribute of the audio object.

Parameters

None.

Usage Notes

None.

Pragmas

PRAGMA RESTRICT_REFERENCES(getFormat, WNDS, WNPS, RNDS, RNPS)

Exceptions

ORDAudioExceptions.AUDIO_FORMAT_IS_NULL

This exception is raised if you call the getFormat( ) method and the value for format is NULL.

Examples

See the example in setProperties( ) on page 4-51.

getNumberOfChannels( )

Format

```
getNumberOfChannels( ) RETURN INTEGER;
```

Description

Returns the value of the numberOfChannels attribute of the audio object.

Parameters

None.

Usage Notes

None.

Pragmas

PRAGMA RESTRICT_REFERENCES(getNumberOfChannels, WNDS, WNPS, RNDS, RNPS)

Exceptions

None.

Examples

See the example in setProperties( ) on page 4-51.

getSampleSize( )

Format

```
getSampleSize( ) RETURN INTEGER;
```

Description

Returns the value of the sampleSize attribute of the audio object.

Parameters

None.

Usage Notes

None.

Pragmas

PRAGMA RESTRICT_REFERENCES(getSampleSize, WNDS, WNPS, RNDS, RNPS)

Exceptions

None.

Examples

See the example in setProperties( ) on page 4-51.

getSamplingRate( )

Format

```
getSamplingRate( ) IN INTEGER;
```

Description

Returns the value of the samplingRate attribute of the audio object. The unit is Hz.

Parameters

None.

Usage Notes

None.

Pragmas

PRAGMA RESTRICT_REFERENCES(getSamplingRate, WNDS, WNPS, RNDS, RNPS)

Exceptions

None.

Examples

See the example in setProperties( ) on page 4-51.

import( )

Format

```
import(ctx IN OUT RAW);
```

Description

Transfers audio data from an external audio data source to the source.localData attribute (of the embedded ORDSource object).

Parameters

ctx
The source plug-in context information. This should be allocated and initialized to NULL. If you are using a user-defined source plug-in, you should call the openSource( ) method. See the introduction to this chapter for more information.

Usage Notes

Use the setSource( ) method to set the source.srcType, source.srcLocation, and source.srcName attributes (of the embedded ORDSource object) for the external source prior to calling the import( ) method.

You must ensure that the directory for the external source location exists or is created before you use this method when the source.srcType attribute value is "file".

After importing data from an external audio data source to a local source (within Oracle Database), the source information remains unchanged (that is, pointing to the source from where the data was imported).

Invoking this method implicitly calls the setUpdateTime( ) and setLocal( ) methods.

This method is invoked at the ORDSource level, which uses the PL/SQL UTL_HTTP package to import media data from an HTTP data source. You can use environment variables to specify the proxy behavior of the UTL_HTTP package. For example, on UNIX, setting the environment variable http_proxy to a URL specifies that the UTL_HTTP package will use that URL as the proxy server for HTTP requests. Setting the no_proxy environment variable to a domain name specifies that the HTTP proxy server will not be used for URLs in the specified domain.

See *PL/SQL Packages and Types Reference* for more information about the UTL_HTTP PL/SQL package.

Pragmas

None.

Exceptions

ORDSourceExceptions.INCOMPLETE_SOURCE_INFORMATION

This exception is raised if you call the import( ) method and the value of the source.srcType attribute is NULL.

ORDSourceExceptions.NULL_SOURCE

This exception is raised if you call the import( ) method and the value of the source.localData attribute is NULL.

ORDSourceExceptions.METHOD_NOT_SUPPORTED

This exception is raised if you call the import( ) method and the import( ) method is not supported by the source plug-in being used.

ORDSourceExceptions.SOURCE_PLUGIN_EXCEPTION

This exception is raised if you call the import( ) method and the source plug-in raises an exception.

See Appendix F for more information about these exceptions.

Examples

Import audio data from an external audio data source into the local source:

```
DECLARE
  obj ORDSYS.ORDAudio;
  ctx RAW(64) := NULL;
BEGIN
   SELECT p.product_audio INTO obj FROM pm.online_media p
     WHERE p.product_id  = 1733 FOR UPDATE;
  DBMS_OUTPUT.PUT_LINE('getting source');
  DBMS_OUTPUT.PUT_LINE('-------------------------');
  -- get source information
  DBMS_OUTPUT.PUT_LINE(obj.getSource());
  -- import data
  obj.import(ctx);
  -- check size
  DBMS_OUTPUT.PUT_LINE('Length is ' || TO_CHAR(obj.getContentLength(ctx)));
  UPDATE pm.online_media p SET p.product_audio = obj WHERE p.product_id = 1733;
  COMMIT;
```

```
END;
/
```

importFrom( )

Format

```
importFrom(ctx                IN OUT RAW,
           source_type        IN VARCHAR2,
           source_location    IN VARCHAR2,
           source_name        IN VARCHAR2);
```

Description

Transfers audio data from the specified external audio data source to the source.localData attribute (of the embedded ORDSource object type) within the database.

Parameters

ctx
The source plug-in context information. This should be allocated and initialized to NULL. If you are using a user-defined source plug-in, you should call the openSource( ) method. See the introduction to this chapter for more information.

source_type
The type of the source audio data.

source_location
The location from which the source audio data is to be imported.

source_name
The name of the source audio data.

Usage Notes

This method is similar to the import( ) method except the source information is specified as parameters to the method instead of separately.

You must ensure that the directory indicated by the source_location parameter exists or is created before you use this method with a source_type parameter value of "file".

After importing data from an external audio data source to a local source (within Oracle Database), the source information (that is, pointing to the source from where the data was imported) is set to the input values.

Invoking this method implicitly calls the setUpdateTime( ) and setLocal( ) methods.

This method is invoked at the ORDSource level, which uses the PL/SQL UTL_HTTP package to import media data from an HTTP data source. You can use environment variables to specify the proxy behavior of the UTL_HTTP package. For example, on UNIX, setting the environment variable http_proxy to a URL specifies that the UTL_HTTP package will use that URL as the proxy server for HTTP requests. Setting the no_proxy environment variable to a domain name specifies that the HTTP proxy server will not be used for URLs in the specified domain.

See *PL/SQL Packages and Types Reference* for more information about the UTL_HTTP PL/SQL package.

Pragmas

None.

Exceptions

ORDSourceExceptions.NULL_SOURCE

This exception is raised if you call the importFrom( ) method and the value of the source.localData attribute is NULL.

ORDSourceExceptions.METHOD_NOT_SUPPORTED

This exception is raised if you call the importFrom( ) method and this method is not supported by the source plug-in being used.

ORDSourceExceptions.SOURCE_PLUGIN_EXCEPTION

This exception is raised if you call the importFrom( ) method and the source plug-in raises an exception.

See Appendix F for more information about these exceptions.

Examples

Import audio data from the specified external data source into the local source:

```
DECLARE
 obj ORDSYS.ORDAudio;
 ctx RAW(64) :=NULL;
BEGIN
```

```
 SELECT p.product_audio INTO obj FROM pm.online_media p
  WHERE p.product_id  = 1729 FOR UPDATE;
 DBMS_OUTPUT.PUT_LINE('setting and getting source');
 DBMS_OUTPUT.PUT_LINE('-------------------------');
 -- set source to a file
 -- import data
 obj.importFrom(ctx,'file','FILE_DIR','birds.wav');
 -- check size
 DBMS_OUTPUT.PUT_LINE('Length is ' || TO_CHAR(obj.getContentLength(ctx)));
 DBMS_OUTPUT.PUT_LINE(obj.getSource());
 UPDATE pm.online_media p SET p.product_audio = obj WHERE p.product_id = 1729;
 COMMIT;
 EXCEPTION
  WHEN ORDSYS.ORDAudioExceptions.METHOD_NOT_SUPPORTED THEN
   DBMS_OUTPUT.PUT_LINE('Source not specified');
 END;
/
```

processAudioCommand( )

Format

```
processAudioCommand(
                    ctx       IN OUT RAW,
                    cmd       IN VARCHAR2,
                    arguments IN VARCHAR2,
                    result    OUT RAW)
RETURN RAW;
```

Description

Lets you send a command and related arguments to the format plug-in for processing.

> **Note:** This method is supported only for user-defined format plug-ins.

Parameters

ctx
The format plug-in context information.

cmd
Any command recognized by the format plug-in.

arguments
The arguments of the command.

result
The result of calling this method returned by the format plug-in.

Usage Notes

Use this method to send any audio commands and their respective arguments to the format plug-in. Commands are not interpreted; they are taken and passed through to a format plug-in to be processed.

To use your user-defined format plug-in, you must set the format attribute to a user-defined format for which you have implemented a plug-in that supports the processAudioCommand( ).

You can extend support to a format that is not understood by the ORDAudio object by preparing an ORDPLUGINS.ORDX_<format>_AUDIO package that supports that format. See *Oracle interMedia User's Guide* for more information.

Pragmas

None.

Exceptions

ORDAudioExceptions.AUDIO_PLUGIN_EXCEPTION

This exception is raised if you call the processAudioCommand( ) method and the audio plug-in raises an exception.

Examples

None.

setAudioDuration( )

Format

setAudioDuration(knownAudioDuration IN INTEGER);

Description

Sets the value of the audioDuration attribute of the audio object.

Parameters

knownAudioDuration
A known audio duration.

Usage Notes

Calling this method implicitly calls the setUpdateTime( ) method.

Pragmas

None.

Exceptions

ORDAudioExceptions.NULL_INPUT_VALUE

This exception is raised if you call the setAudioDuration( ) method and the value for the knownAudioDuration parameter is NULL.

Examples

See the example in setFormat( ) on page 4-45.

setCompressionType( )

Format

setCompressionType(knownCompressionType IN VARCHAR2);

Description

Sets the value of the compressionType attribute of the audio object.

Parameters

knownCompressionType
A known compression type.

Usage Notes

The value of the compressionType always matches that of the encoding value because in many audio formats, encoding and compression type are tightly integrated. See Appendix A for more information.

Calling this method implicitly calls the setUpdateTime( ) method.

Pragmas

None.

Exceptions

ORDAudioExceptions.NULL_INPUT_VALUE

This exception is raised if you call the setCompressionType( ) method and the value for the knownCompressionType parameter is NULL.

Examples

See the example in setFormat( ) on page 4-45.

setDescription( )

Format

setDescription (user_description IN VARCHAR2);

Description

Sets the description of the audio data.

Parameters

user_description
The description of the audio data.

Usage Notes

Each audio object may need a description to help some client applications. For example, a Web-based client can show a list of audio descriptions from which a user can select one to access the audio data.

Web-access components and other client components provided with *inter*Media make use of this description attribute to present audio data to users.

Calling this method implicitly calls the setUpdateTime( ) method.

Pragmas

None.

Exceptions

None.

Examples

Set the description attribute for some audio data:

```
DECLARE
  obj ORDSYS.ORDAudio;
BEGIN
  SELECT p.product_audio INTO obj FROM pm.online_media p
    WHERE p.product_id  = 1733 FOR UPDATE;
  DBMS_OUTPUT.PUT_LINE('writing new title');
  DBMS_OUTPUT.PUT_LINE('-------------');
```

```
  obj.setDescription('This is audio for product 1733');
  DBMS_OUTPUT.PUT_LINE(obj.getDescription);
  UPDATE pm.online_media p SET p.product_audio = obj WHERE p.product_id = 1733;
  COMMIT;
END;
/
```

setEncoding( )

Format

```
setEncoding(knownEncoding IN VARCHAR2);
```

Description

Sets the value of the encoding attribute of the audio object.

Parameters

knownEncoding
A known encoding type.

Usage Notes

The value of encoding always matches that of the compressionType value because in many audio formats, encoding and compression type are tightly integrated. See Appendix A for more information.

Calling this method implicitly calls the setUpdateTime( ) method.

Pragmas

None.

Exceptions

ORDAudioExceptions.NULL_INPUT_VALUE

This exception is raised if you call the setEncoding( ) method and the value for the knownEncoding parameter is NULL.

Examples

See the example in setFormat( ) on page 4-45.

setFormat( )

Format

setFormat(knownFormat IN VARCHAR2);

Description

Sets the format attribute of the audio object.

Parameters

knownFormat
The known format of the audio data to be set in the audio object.

Usage Notes

Calling this method implicitly calls the setUpdateTime( ) method.

Pragmas

None.

Exceptions

ORDAudioExceptions.NULL_INPUT_VALUE

This exception is raised if you call the setFormat( ) method and the value for the knownFormat parameter is NULL.

Examples

Set the format (and other attributes) for some audio data:

```
DECLARE
 obj ORDSYS.ORDAudio;
BEGIN
 SELECT p.product_audio INTO obj FROM pm.online_media p
  WHERE p.product_id  = 1733 FOR UPDATE;
 obj.setFormat('AUFF');
 obj.setEncoding('MULAW');
 obj.setNumberOfChannels(1);
 obj.setSamplingRate(8);
 obj.setSampleSize(8);
 obj.setCompressionType('8BITMONOAUDIO');
```

```
 obj.setAudioDuration(16);
 DBMS_OUTPUT.PUT_LINE('format: ' || obj.getformat);
 DBMS_OUTPUT.PUT_LINE('encoding: ' || obj.getEncoding);
 DBMS_OUTPUT.PUT_LINE(
                     'numberOfChannels: ' || TO_CHAR(obj.getNumberOfChannels));
 DBMS_OUTPUT.PUT_LINE('samplingRate: ' || TO_CHAR(obj.getSamplingRate));
 DBMS_OUTPUT.PUT_LINE('sampleSize: ' || TO_CHAR(obj.getSampleSize));
 DBMS_OUTPUT.PUT_LINE('compressionType : ' || obj.getCompressionType);
 DBMS_OUTPUT.PUT_LINE('audioDuration: ' || TO_CHAR(obj.getAudioDuration));
 COMMIT;
 EXCEPTION
  WHEN ORDSYS.ORDAudioExceptions.NULL_INPUT_VALUE THEN
   DBMS_OUTPUT.PUT_LINE('ORDAudioExceptions.NULL_INPUT_VALUE caught');
  WHEN OTHERS THEN
   DBMS_OUTPUT.PUT_LINE('EXCEPTION caught');
END;
/
```

setKnownAttributes( )

Format

```
setKnownAttributes(
                    knownFormat              IN VARCHAR2,
                    knownEncoding            IN VARCHAR2,
                    knownNumberOfChannels IN INTEGER,
                    knownSamplingRate        IN INTEGER,
                    knownSampleSize          IN INTEGER,
                    knownCompressionType  IN VARCHAR2,
                    knownAudioDuration       IN INTEGER);
```

Description

Sets the known audio attributes for the audio object.

Parameters

knownFormat
The known format.

knownEncoding
The known encoding type.

knownNumberOfChannels
The known number of channels.

knownSamplingRate
The known sampling rate.

knownSampleSize
The known sample size.

knownCompressionType
The known compression type.

knownAudioDuration
The known audio duration.

Usage Notes

Calling this method implicitly calls the setUpdateTime( ) method.

Pragmas

None.

Exceptions

None.

Examples

Set the known attributes for the audio data:

```
DECLARE
 obj ORDSYS.ORDAudio;
BEGIN
 SELECT p.product_audio INTO obj FROM pm.online_media p
  WHERE p.product_id  = 1733 FOR UPDATE;
 obj.setKnownAttributes('AUFF','MULAW', 1, 8, 8, '8BITMONOAUDIO',16);
 DBMS_OUTPUT.PUT_LINE('format: ' || obj.getformat());
 DBMS_OUTPUT.PUT_LINE('encoding: ' || obj.getEncoding());
 DBMS_OUTPUT.PUT_LINE(
                   'numberOfChannels: ' || TO_CHAR(obj.getNumberOfChannels()));
 DBMS_OUTPUT.PUT_LINE('samplingRate: ' || TO_CHAR(obj.getSamplingRate()));
 DBMS_OUTPUT.PUT_LINE('sampleSize: ' || TO_CHAR(obj.getSampleSize()));
 DBMS_OUTPUT.PUT_LINE('compressionType : ' || obj.getCompressionType());
 DBMS_OUTPUT.PUT_LINE('audioDuration: ' || TO_CHAR(obj.getAudioDuration()));
 UPDATE pm.online_media p SET p.product_audio = obj
   WHERE  p.product_id = 1733;
 COMMIT;
 EXCEPTION
  WHEN ORDSYS.ORDAudioExceptions.METHOD_NOT_SUPPORTED THEN
   DBMS_OUTPUT.PUT_LINE('ORDAudioExceptions.METHOD_NOT_SUPPORTED caught');
  WHEN OTHERS THEN
   DBMS_OUTPUT.PUT_LINE('EXCEPTION caught');
END;
/
```

setNumberOfChannels( )

Format

setNumberOfChannels(knownNumberOfChannels IN INTEGER);

Description

Sets the value of the numberOfChannels attribute for the audio object.

Parameters

knownNumberOfChannels
A known number of channels.

Usage Notes

Calling this method implicitly calls the setUpdateTime( ) method.

Pragmas

None.

Exceptions

ORDAudioExceptions.NULL_INPUT_VALUE

This exception is raised if you call the setNumberOfChannels( ) method and the value for the knownNumberOfChannels parameter is NULL.

Examples

See the example in setFormat( ) on page 4-45.

setProperties( )

Format

```
setProperties(ctx                IN OUT RAW,
              setComments IN BOOLEAN);
```

Description

Reads the audio data to get the values of the object attributes and then stores them in the object attributes. This method sets the properties for each of the following attributes of the audio data for which values are available: compression type, duration, encoding type, format, mime type, number of channels, sampling rate, and sample size. It populates the comments field of the object with a rich set of format and application properties in XML form if the value of the setComments parameter is TRUE.

Parameters

ctx
The format plug-in context information.

setComments
A Boolean value that indicates whether or not the comments field of the object is populated. If the value is TRUE, then the comments field of the object is populated with a rich set of format and application properties of the audio object in XML form; otherwise, if the value is FALSE, the comments field of the object remains unpopulated. The default value is FALSE.

Usage Notes

If the property cannot be extracted from the media source, then the respective attribute is set to the NULL value.

If the format attribute is set to the NULL value before calling this method, then the setProperties( ) method uses the default format plug-in; otherwise, it uses the plug-in specified by the format.

Pragmas

None.

Exceptions

ORDAudioExceptions.AUDIO_PLUGIN_EXCEPTION

This exception is raised if you call the setProperties( ) method and the audio plug-in raises an exception.

Example

Set the property information for known audio attributes:

```
DECLARE
 obj ORDSYS.ORDAudio;
 ctx RAW(64) :=NULL;
BEGIN
 SELECT p.product_audio INTO obj FROM pm.online_media p
  WHERE p.product_id = 1729 FOR UPDATE;
 obj.setProperties(ctx,FALSE);
 DBMS_OUTPUT.PUT_LINE('format: ' || obj.getformat);
 DBMS_OUTPUT.PUT_LINE('encoding: ' || obj.getEncoding);
 DBMS_OUTPUT.PUT_LINE(
                 'numberOfChannels: ' || TO_CHAR(obj.getNumberOfChannels));
 DBMS_OUTPUT.PUT_LINE('samplingRate: ' || TO_CHAR(obj.getSamplingRate));
 DBMS_OUTPUT.PUT_LINE('sampleSize: ' || TO_CHAR(obj.getSampleSize));
 UPDATE  pm.online_media p set p.product_audio = obj
   WHERE  p.product_id = 1733;
 COMMIT;
 EXCEPTION
  WHEN ORDSYS.ORDAudioExceptions.METHOD_NOT_SUPPORTED THEN
   DBMS_OUTPUT.PUT_LINE('ORDAudioExceptions.METHOD_NOT_SUPPORTED caught');
  WHEN OTHERS THEN
   DBMS_OUTPUT.PUT_LINE('EXCEPTION caught');
END;
/
```

setSamplingRate( )

Format

setSamplingRate(knownSamplingRate IN INTEGER);

Description

Sets the value of the samplingRate attribute of the audio object. The unit is Hz.

Parameters

knownSamplingRate
A known sampling rate.

Usage Notes

Calling this method implicitly calls the setUpdateTime( ) method.

Pragmas

None.

Exceptions

ORDAudioExceptions.NULL_INPUT_VALUE

This exception is raised if you call the setSamplingRate( ) method and the value for the knownSamplingRate parameter is NULL.

Examples

See the example in setFormat( ) on page 4-45.

setSampleSize( )

Format

setSampleSize(knownSampleSize IN INTEGER);

Description

Sets the value of the sampleSize attribute of the audio object.

Parameters

knownSampleSize
A known sample size.

Usage Notes

Calling this method implicitly calls the setUpdateTime( ) method.

Pragmas

None.

Exceptions

ORDAudioExceptions.NULL_INPUT_VALUE

This exception is raised if you call the setSampleSize( ) method and the value for the knownSampleSize parameter is NULL.

Examples

See the example in setFormat( ) on page 4-45.

5

ORDDoc

Oracle *inter*Media (*"inter*Media") contains the following information about the ORDDoc object type:

The examples in this chapter use the ONLINE_MEDIA table in the Product Media sample schema. To replicate the examples on your own computer, you should begin with the examples shown in the reference pages for the ORDDoc constructors and the import( ) and importFrom( ) methods. Substitute files you have for the ones shown in the examples. In addition, for a user "ron" to use the examples, the following statements must be issued before ron executes the examples, where "/mydir/work" is the directory where ron will find the image, audio, and video data:

```
CONNECT /as sysdba
CREATE OR REPLACE DIRECTORY FILE_DIR as '/mydir/work';
GRANT READ ON DIRECTORY FILE_DIR TO PUBLIC WITH GRANT OPTION;
```

See *Oracle Database Sample Schemas* for information on the sample schemas.

Note: If you manipulate the media data itself (by either directly modifying the BLOB or changing the external source), then you must ensure that the object attributes stay synchronized and the update time is modified; otherwise, the object attributes will not match the media data.

Methods invoked at the ORDSource level that are handed off to the source plug-in for processing have ctx (RAW) as the first argument. Before calling any of these methods for the first time, the client should allocate the ctx structure, initialize it to NULL, and invoke the openSource( ) method. At this point, the source plug-in can initialize context for this client. When processing is complete, the client should invoke the closeSource( ) method.

Methods invoked from a source plug-in call have the first argument as ctx (RAW).

Methods invoked at the ORDDoc level that are handed off to the format plug-in for processing have ctx (RAW) as the first argument. Before calling any of these methods for the first time, the client should allocate the ctx structure and initialize it to NULL.

Note: In the current release, none of the plug-ins provided by Oracle and not all source or format plug-ins will use the ctx argument, but if you code as previously described, your application should work with current or future source or format plug-ins.

You should use any of the individual set methods to set the attribute value for an object for formats not natively supported; otherwise, for formats natively supported, use the setProperties( ) method to populate the attributes of the object or write a format plug-in.

ORDDoc Object Type

The ORDDoc object type supports the storage and management of any media data including image, audio, and video. This object type is defined as follows:

```
CREATE OR REPLACE TYPE ORDDoc
AS OBJECT
(
  -- ATTRIBUTES
source              ORDSource,
format              VARCHAR(80),
mimeType            VARCHAR(80),
contentLength       INTEGER,
comments            CLOB,

-- METHODS
-- CONSTRUCTORS
--
STATIC FUNCTION init( ) RETURN ORDDoc,
STATIC FUNCTION init(srcType     IN VARCHAR2,
                     srcLocation IN VARCHAR2,
                     srcName     IN VARCHAR2) RETURN ORDDoc,
-- Methods associated with the mimeType attribute
MEMBER FUNCTION getMimeType RETURN VARCHAR2,
PRAGMA RESTRICT_REFERENCES(getMimeType, WNDS, WNPS, RNDS, RNPS),
MEMBER PROCEDURE setMimeType(mime IN VARCHAR2),

-- Methods associated with the date attribute
MEMBER FUNCTION getUpdateTime RETURN DATE,
PRAGMA RESTRICT_REFERENCES(getUpdateTime, WNDS, WNPS, RNDS, RNPS),
MEMBER PROCEDURE setUpdateTime(current_time DATE),

-- Methods associated with the format attribute
MEMBER FUNCTION getFormat RETURN VARCHAR2,
PRAGMA RESTRICT_REFERENCES(getFormat, WNDS, WNPS, RNDS, RNPS),
MEMBER PROCEDURE setFormat(format IN VARCHAR2),

-- Methods associated with the source attribute
MEMBER FUNCTION  isLocal RETURN BOOLEAN,
PRAGMA RESTRICT_REFERENCES(isLocal, WNDS, WNPS, RNDS, RNPS),
MEMBER PROCEDURE  setLocal,
MEMBER PROCEDURE  clearLocal,
```

```
MEMBER PROCEDURE setSource(source_type     IN VARCHAR2,
                           source_location IN VARCHAR2,
                           source_name     IN VARCHAR2),
MEMBER FUNCTION getSource RETURN VARCHAR2,
PRAGMA RESTRICT_REFERENCES(getSource, WNDS, WNPS, RNDS, RNPS),

MEMBER FUNCTION getSourceType RETURN VARCHAR2,
PRAGMA RESTRICT_REFERENCES(getSourceType, WNDS, WNPS, RNDS, RNPS),

MEMBER FUNCTION getSourceLocation RETURN VARCHAR2,
PRAGMA RESTRICT_REFERENCES(getSourceLocation, WNDS, WNPS, RNDS, RNPS),

MEMBER FUNCTION getSourceName RETURN VARCHAR2,
PRAGMA RESTRICT_REFERENCES(getSourceName, WNDS, WNPS, RNDS, RNPS),

MEMBER PROCEDURE setProperties(ctx         IN OUT RAW,
                               setComments IN BOOLEAN),

MEMBER FUNCTION getBFILE RETURN BFILE,
PRAGMA RESTRICT_REFERENCES(getBFILE, WNDS, WNPS, RNDS, RNPS),

MEMBER PROCEDURE import(ctx      IN OUT RAW,
                       set_prop IN BOOLEAN),
MEMBER PROCEDURE importFrom(ctx             IN OUT RAW,
                            source_type     IN VARCHAR2,
                            source_location IN VARCHAR2,
                            source_name     IN VARCHAR2
                            set_prop        IN BOOLEAN),
MEMBER PROCEDURE export(ctx             IN OUT RAW,
                        source_type     IN VARCHAR2,
                        source_location IN VARCHAR2,
                        source_name     IN VARCHAR2),

MEMBER FUNCTION openSource(userArg IN RAW, ctx OUT RAW) RETURN INTEGER,
MEMBER FUNCTION closeSource(ctx IN OUT RAW) RETURN INTEGER,
MEMBER FUNCTION trimSource(ctx     IN OUT RAW,
                           newlen  IN INTEGER) RETURN INTEGER,
MEMBER PROCEDURE readFromSource(ctx      IN OUT RAW,
                                startPos IN INTEGER,
                                numBytes IN OUT INTEGER,
                                buffer   OUT RAW),
MEMBER PROCEDURE writeToSource(ctx      IN OUT RAW,
                               startPos IN INTEGER,
                               numBytes IN OUT INTEGER,
                               buffer   IN RAW),
```

```
MEMBER FUNCTION getContentLength RETURN INTEGER,
PRAGMA RESTRICT_REFERENCES(getContentLength, WNDS, WNPS, RNDS, RNPS),

MEMBER PROCEDURE getContentInLob(ctx      IN OUT RAW,
                                 dest_lob  IN OUT NOCOPY BLOB,
                                 mimeType  OUT VARCHAR2,
                                 format    OUT VARCHAR2),

MEMBER FUNCTION getContent RETURN BLOB,
PRAGMA RESTRICT_REFERENCES(getContent, WNDS, WNPS, RNDS, RNPS),

MEMBER PROCEDURE deleteContent,

MEMBER FUNCTION processSourceCommand(ctx       IN OUT RAW,
                                     cmd       IN VARCHAR2,
                                     arguments IN VARCHAR2,
                                     result    OUT RAW)
                RETURN RAW

);
```

where:

- source: the ORDSource where the media data is found.
- format: the format in which the media data is stored.
- mimeType: the MIME type information.
- contentLength: the length of the media data stored in the source.
- comments: the metadata information of the media object.

> **Note:** The comments attribute is populated by the setProperties( ) method when the setComments parameter is TRUE and by Oracle *inter*Media Annotator for natively supported formats. Oracle Corporation recommends that you not write to this attribute directly.

ORDDoc Constructors

This section describes the ORDDoc constructor functions, which are the following:

init( ) for ORDDoc

Format

init( ) RETURN ORDDoc;

Description

Initializes instances of the ORDDoc object type.

Parameters

None.

Pragmas

None.

Exceptions

None.

Usage Notes

This constructor is a static method that initializes all the ORDDoc attributes to NULL with the following exceptions:

- source.updateTime is set to SYSDATE
- source.local is set to 1 (local)
- source.localData is set to empty_blob

You should begin using the init( ) method as soon as possible so you can more easily initialize the ORDDoc object type, especially if the ORDDoc type evolves and attributes are added in a future release. INSERT statements left unchanged using the default constructor (which initializes each object attribute), will fail under these circumstances.

Examples

Initialize the ORDDoc object attributes:

```
BEGIN
  INSERT INTO pm.online_media (product_id, product_testimonials)
     VALUES (2808,ORDSYS.ORDDoc.init());
```

```
  COMMIT;
END;
/
```

init(srcType,srcLocation,srcName) for ORDDoc

Format

```
init(srcType     IN VARCHAR2,
   srcLocation IN VARCHAR2,
   srcName     IN VARCHAR2)
 RETURN ORDDoc;
```

Description

Initializes instances of the ORDDoc object type.

Parameters

srcType
The source type of the media data.

srcLocation
The source location of the media data.

srcName
The source name of the media data.

Pragmas

None.

Exceptions

None.

Usage Notes

This constructor is a static method that initializes all the ORDDoc attributes to NULL with the following exceptions:

- source.updateTime is set to SYSDATE
- source.local is set to 0
- source.localData is set to empty_blob

- source.srcType is set to the input value
- source.srcLocation is set to the input value
- source.srcName is set to the input value

You should begin using the init( ) method as soon as possible to allow you to more easily initialize the ORDDoc object type, especially if the ORDDoc type evolves and attributes are added in a future release. INSERT statements left unchanged using the default constructor (which initializes each object attribute), will fail under these circumstances.

Examples

Initialize the ORDDoc object attributes:

```
BEGIN
INSERT INTO pm.online_media (product_id, product_testimonials)
   VALUES (2999, ORDSYS.ORDDoc.init('file', 'FILE_DIR', 'modem.jpg'));
END;
/
```

ORDDoc Methods

This section presents reference information on the *inter*Media methods used specifically for media data manipulation.

Chapter 3 presents reference information on the *inter*Media methods that are common to ORDAudio, ORDDoc, ORDImage, and ORDVideo. Use the methods presented in both chapters to get and set attributes, and to perform metadata extractions.

For more information on object types and methods, see *Oracle Database Concepts*.

getContentInLob( )

Format

```
getContentInLob(
                ctx          IN OUT RAW,
                dest_lob     IN OUT NOCOPY BLOB,
                mimeType     OUT VARCHAR2,
                format       OUT VARCHAR2);
```

Description

Copies data from a data source into the specified BLOB. The BLOB must not be the BLOB in the source.localData attribute (of the embedded ORDSource object).

Parameters

ctx
The source plug-in context information.

dest_lob
The LOB in which to receive data.

mimeType
The MIME type of the data; this may or may not be returned.

format
The format of the data; this may or may not be returned.

Usage Notes

None.

Pragmas

None.

Exceptions

ORDSourceExceptions.METHOD_NOT_SUPPORTED

This exception is raised if you call the getContentInLob( ) method and this method is not supported by the source plug-in being used.

ORDSourceExceptions.SOURCE_PLUGIN_EXCEPTION

This exception is raised if you call the getContentInLob( ) method and the source plug-in raises an exception.

See Appendix F for more information about these exceptions.

Examples

Get data from a data source and put it into the specified BLOB:

```
DECLARE
 obj ORDSYS.ORDDoc;
 tempBLob BLOB;
 mimeType VARCHAR2(4000);
 format VARCHAR2(31);
 ctx RAW(64) :=NULL;
BEGIN
 SELECT product_testimonials INTO obj FROM pm.online_media
    WHERE product_id = 2808 ;
 IF (obj.isLocal()) THEN
   DBMS_OUTPUT.put_line('Local is true');
 END IF;
 DBMS_LOB.CREATETEMPORARY(tempBLob, true, 10);
 obj.getContentInLob(ctx,tempBLob, mimeType,format);
 DBMS_OUTPUT.PUT_LINE('Length: ' || TO_CHAR(DBMS_LOB.getLength(tempBLob)));
 EXCEPTION
  WHEN ORDSYS.ORDSourceExceptions.METHOD_NOT_SUPPORTED THEN
   DBMS_OUTPUT.put_line('ORDSourceExceptions.METHOD_NOT_SUPPORTED caught');
  WHEN OTHERS THEN
   DBMS_OUTPUT.put_line('EXCEPTION caught');
END;
/
```

getContentLength( )

Format

getContentLength( ) RETURN INTEGER;

Description

Returns the length of the media data content stored in the source.

Parameters

None.

Usage Notes

None.

Pragmas

PRAGMA RESTRICT_REFERENCES(getContentLength, WNDS, WNPS, RNDS, RNPS)

Exceptions

None.

Examples

```
DECLARE
 obj ORDSYS.ORDDoc;
BEGIN
 SELECT product_testimonials INTO obj FROM pm.online_media
   WHERE product_id = 2808 ;
 IF (obj.isLocal()) THEN
   DBMS_OUTPUT.put_line('Local is true');
 END IF;
 DBMS_OUTPUT.PUT_LINE('Content length is ' || TO_CHAR(obj.getContentLength));
END;
/
```

getFormat( )

Format

```
getFormat( ) RETURN VARCHAR2;
```

Description

Returns the value of the format attribute of the media object.

Parameters

None.

Usage Notes

None.

Pragmas

PRAGMA RESTRICT_REFERENCES(getFormat, WNDS, WNPS, RNDS, RNPS)

Exceptions

ORDDocExceptions.INVALID_FORMAT_TYPE

This exception is raised if you call the getFormat( ) method and the value of the format attribute is NULL.

Examples

See the example in setFormat( ) on page 5-22.

import( )

Format

```
import(ctx        IN OUT RAW
       set_prop IN BOOLEAN);
```

Description

Transfers media data from an external media data source to the source.localData attribute (of the embedded ORDSource object) within the database.

Parameters

ctx
The source plug-in context information. This should be allocated and initialized to NULL. If you are using a user-defined source plug-in, you should call the openSource( ) method. See the introduction to this chapter for more information.

set_prop
A value that determines whether the setProperties( ) method is called. If the value of this parameter is TRUE, then the setProperties( ) method is called to read the media data to get the values of the object attributes and store them in the object attributes; otherwise, if the value is FALSE, the set Properties( ) method is not called. The default value is FALSE.

Usage Notes

Use the setSource( ) method to set the source.srcType, source.srcLocation, and source.Name attributes (of the embedded ORDSource object) for the external media data source prior to calling the import( ) method.

After importing data from an external media data source to a local source (within Oracle Database), the source information remains unchanged (that is, pointing to the source from where the data was imported).

Invoking this method implicitly calls the setUpdateTime( ) and setLocal( ) methods.

This method is invoked at the ORDSource level, which uses the PL/SQL UTL_HTTP package to import media data from an HTTP data source. You can use environment variables to specify the proxy behavior of the UTL_HTTP package. For example, on UNIX, setting the environment variable http_proxy to a URL specifies that the UTL_HTTP package will use that URL as the proxy server for HTTP

requests. Setting the no_proxy environment variable to a domain name specifies that the HTTP proxy server will not be used for URLs in the specified domain.

See *PL/SQL Packages and Types Reference* for more information about the UTL_HTTP PL/SQL package.

Pragmas

None.

Exceptions

ORDSourceExceptions.INCOMPLETE_SOURCE_INFORMATION

This exception is raised if you call the import( ) method and the value of the source.srcType attribute is NULL.

ORDSourceExceptions.NULL_SOURCE

This exception is raised if you call the import( ) method and the value of the source.localData attribute is NULL.

ORDSourceExceptions.METHOD_NOT_SUPPORTED

This exception is raised if you call the import( ) method and the import( ) method is not supported by the source plug-in being used.

ORDSourceExceptions.SOURCE_PLUGIN_EXCEPTION

This exception is raised if you call the import( ) method and the source plug-in raises an exception.

ORDSYS.ORDDocExceptions.DOC_PLUGIN_EXCEPTION

This exception is raised if you call the import( ) method and the setProperties( ) method raises an exception from within the media plug-in.

See Appendix F for more information about these exceptions.

Examples

Import media data from an external media data source into the local source:

```
DECLARE
  obj ORDSYS.ORDDoc;
  ctx RAW(64) :=NULL;
BEGIN
  SELECT product_testimonials INTO obj FROM pm.online_media
    WHERE product_id = 2808 FOR UPDATE;
```

```
  DBMS_OUTPUT.PUT_LINE('setting and getting source');
  DBMS_OUTPUT.PUT_LINE('--------------------------');
  -- set source to a file
  obj.setSource('file','FILE_DIR','printer.rm');
  -- get source information
  DBMS_OUTPUT.PUT_LINE(obj.getSource());
  -- import data
  obj.import(ctx,FALSE);
  -- check size
  DBMS_OUTPUT.PUT_LINE('Length:' ||TO_CHAR(DBMS_LOB.getLength(obj.getContent)));
  UPDATE pm.online_media SET product_testimonials=obj WHERE product_id=2808;
  COMMIT;
  EXCEPTION
    WHEN ORDSYS.ORDSourceExceptions.METHOD_NOT_SUPPORTED THEN
      DBMS_OUTPUT.PUT_LINE('ORDSourceExceptions.METHOD_NOT_SUPPORTED caught');
    WHEN ORDSYS.ORDDocExceptions.DOC_PLUGIN_EXCEPTION THEN
      DBMS_OUTPUT.put_line('DOC PLUGIN EXCEPTION caught');
END;
/
```

importFrom( )

Format

```
importFrom(ctx                IN OUT RAW,
           source_type        IN VARCHAR2,
           source_location    IN VARCHAR2,
           source_name        IN VARCHAR2
           set_prop           IN BOOLEAN);
```

Description

Transfers media data from the specified external media data source to the source.localData attribute (of the embedded ORDSource object) within the database).

Parameters

ctx
The source plug-in context information. This should be allocated and initialized to NULL. If you are using a user-defined source plug-in, you should call the openSource( ) method. See the introduction to this chapter for more information.

source_type
The type of the source media data.

source_location
The location from which the source media data is to be imported.

source_name
The name of the source media data.

set_prop
A value that determines whether the setProperties( ) method is called. If the value of this parameter is TRUE, then the setProperties( ) method is called to read the media data to get the values of the object attributes and store them in the object attributes; otherwise, if the value is FALSE, the set Properties( ) method is not called. The default value is FALSE.

Usage Notes

This method is similar to the import( ) method except the source information is specified as parameters to the method instead of separately.

You must ensure that the directory indicated by the source_location parameter exists or is created before you use this method.

After importing data from an external media data source to a local source (within Oracle Database), the source information (that is, pointing to the source from where the data was imported) is set to the input values.

Invoking this method implicitly calls the setUpdateTime( ) and setLocal( ) methods.

This method is invoked at the ORDSource level, which uses the PL/SQL UTL_HTTP package to import media data from an HTTP data source. You can use environment variables to specify the proxy behavior of the UTL_HTTP package. For example, on UNIX, setting the environment variable http_proxy to a URL specifies that the UTL_HTTP package will use that URL as the proxy server for HTTP requests. Setting the no_proxy environment variable to a domain name specifies that the HTTP proxy server will not be used for URLs in the specified domain.

See *PL/SQL Packages and Types Reference* for more information about the UTL_HTTP PL/SQL package.

Pragmas

None.

Exceptions

ORDSourceExceptions.INCOMPLETE_SOURCE_INFORMATION

This exception is raised if you call the importFrom( ) method and the value of the source_type parameter is NULL.

ORDSourceExceptions.METHOD_NOT_SUPPORTED

This exception is raised if you call the importFrom( ) method and this method is not supported by the source plug-in being used.

ORDSourceExceptions.SOURCE_PLUGIN_EXCEPTION

This exception is raised if you call the importFrom( ) method and the source plug-in raises an exception.

ORDSYS.ORDDocExceptions.DOC_PLUGIN_EXCEPTION

This exception is raised if you call the importFrom( ) method and the setProperties( ) method raises an exception from within the media plug-in.

See Appendix F for more information about these exceptions.

Examples

Import media data from the specified external data source into the local source:

```
DECLARE
  obj ORDSYS.ORDDoc;
  ctx RAW(64) :=NULL;
BEGIN
  SELECT product_testimonials INTO obj FROM pm.online_media
    WHERE product_id=2999 FOR UPDATE;
  DBMS_OUTPUT.PUT_LINE('setting and getting source');
  DBMS_OUTPUT.PUT_LINE('--------------------------');
  -- set source to a file
  -- import data
  obj.importFrom(ctx,'file','FILE_DIR','modem.jpg',FALSE);
  -- check size
  DBMS_OUTPUT.PUT_LINE('Length: '||TO_CHAR(DBMS_LOB.GETLENGTH(obj.getContent)));
  DBMS_OUTPUT.PUT_LINE(obj.getSource());
  UPDATE pm.online_media SET product_testimonials=obj WHERE product_id=2999;
  COMMIT;
  EXCEPTION
    WHEN ORDSYS.ORDSourceExceptions.METHOD_NOT_SUPPORTED THEN
      DBMS_OUTPUT.PUT_LINE('ORDSourceExceptions.METHOD_NOT_SUPPORTED caught');
    WHEN ORDSYS.ORDDocExceptions.DOC_PLUGIN_EXCEPTION THEN
      DBMS_OUTPUT.put_line('DOC PLUGIN EXCEPTION caught');
END;
/
```

setFormat( )

Format

setFormat(knownFormat IN VARCHAR2);

Description

Sets the format attribute of the media object.

Parameters

knownFormat
The known format of the data to be set in the corresponding media object.

Usage Notes

Calling this method implicitly calls the setUpdateTime( ) method.

Pragmas

None.

Exceptions

ORDDocExceptions.NULL_INPUT_VALUE

This exception is raised if you call the setFormat( ) method and the value for the knownFormat parameter is NULL.

Examples

Set the format for some media data:

```
DECLARE
  obj ORDSYS.ORDDoc;
BEGIN
  SELECT product_testimonials INTO obj FROM pm.online_media
    WHERE product_id = 2808 FOR UPDATE;
  obj.setFormat('PDF');
  DBMS_OUTPUT.put_line('format: ' || obj.getformat());
  COMMIT;
  EXCEPTION
   WHEN ORDSYS.ORDDocExceptions.NULL_INPUT_VALUE THEN
    DBMS_OUTPUT.put_line('ORDDocExceptions.NULL_INPUT_VALUE caught');
   WHEN OTHERS THEN
```

```
    DBMS_OUTPUT.put_line('EXCEPTION caught');
END;
/
```

setProperties( )

Format

```
setProperties(ctx          IN OUT RAW,
              setComments IN BOOLEAN);
```

Description

Reads the media data to get the values of the object attributes and then stores them in the object attributes. This method sets the properties for the following attributes of the media data: format, MIME type, and content length. It populates the comments field of the object with an extensive set of format and application properties in XML form if the value of the setComments parameter is TRUE.

> **Note:** This method works for only natively supported audio, image, and video formats. See Appendix A, Appendix B, and Appendix C for information on natively supported audio, image, and video formats.

Parameters

ctx
The format plug-in context information.

setComments
A Boolean value that indicates whether or not the comments field of the object is populated. If the value is TRUE, then the comments field of the object is populated with an extensive set of format and application properties of the media object in XML form; otherwise, if the value is FALSE, the comments field of the object remains unpopulated. The default value is FALSE.

Usage Notes

If the property cannot be extracted from the media source, then the respective attribute is set to NULL.

If the format attribute is set to NULL, then the setProperties( ) method uses the default format plug-in; otherwise, it uses the plug-in specified by the format.

Pragmas

None.

Exceptions

ORDDocExceptions.DOC_PLUGIN_EXCEPTION

This exception is raised if you call the setProperties( ) method and the media plug-in raises an exception.

Examples

Set the property information for known media attributes:

```
DECLARE
  obj ORDSYS.ORDDoc;
  ctx RAW(64) :=NULL;
BEGIN
  SELECT product_testimonials INTO obj FROM pm.online_media
    WHERE product_id = 2808 FOR UPDATE;
  obj.setProperties(ctx,FALSE);
  DBMS_OUTPUT.put_line('format: ' || obj.getformat());
  UPDATE  pm.online_media SET product_testimonials = obj
    WHERE  product_id=2808;
  COMMIT;
  EXCEPTION
   WHEN ORDSYS.ORDDocExceptions.DOC_PLUGIN_EXCEPTION THEN
    DBMS_OUTPUT.put_line('DOC PLUGIN EXCEPTION caught');
   WHEN OTHERS THEN
    DBMS_OUTPUT.put_line('EXCEPTION caught');
END;
/
```

Set the property information for known media attributes and store the format and application properties in the comments attribute. Create an extensible index on the contents of the comments attribute using Oracle Text:

```
DECLARE
  obj ORDSYS.ORDDoc;
  ctx RAW(64) :=NULL;
BEGIN
  SELECT product_testimonials INTO obj FROM pm.online_media
    WHERE product_id = 2999 FOR UPDATE;
  obj.setProperties(ctx,TRUE);
  DBMS_OUTPUT.put_line('format: ' || obj.getformat());
   UPDATE  pm.online_media SET product_testimonials = obj
    WHERE  product_id=2999;
  COMMIT;
```

```
  EXCEPTION
   WHEN ORDSYS.ORDDocExceptions.DOC_PLUGIN_EXCEPTION THEN
    DBMS_OUTPUT.put_line('DOC PLUGIN EXCEPTION caught');
   WHEN OTHERS THEN
    DBMS_OUTPUT.put_line('EXCEPTION caught');
END;
/
CONNECT / AS SYSDBA
-- Must have Oracle Text installed on your system.
CREATE INDEX mediaindex ON pm.online_media(product_testimonials.comments)
  INDEXTYPE IS ctxsys.context;
```

6

ORDImage and ORDImageSignature

Oracle *inter*Media (*"inter*Media") contains the following information about the ORDImage object type and the ORDImageSignature object type:

The examples in this chapter use the ONLINE_MEDIA table in the Product Media sample schema. To replicate the examples on your own computer, you should begin with the examples shown in the reference pages for the ORDImage and ORDImageSignature constructors and the import( ) and importFrom( ) methods. Substitute image files you have for the ones shown in the examples. In addition, for a user "ron" to use the examples, the following statements must be issued before ron executes the examples, where "/mydir/work" is the directory where ron will find the image data:

```
CONNECT /as sysdba
CREATE OR REPLACE DIRECTORY FILE_DIR as '/mydir/work';
GRANT READ ON DIRECTORY FILE_DIR TO PUBLIC WITH GRANT OPTION;
```

See *Oracle Database Sample Schemas* for information on the sample schemas.

Note: If you manipulate the image data itself (by either directly modifying the BLOB or changing the external source), then you must ensure that the object attributes stay synchronized and the update time is modified; otherwise, the object attributes will not match the image data.

Methods invoked at the ORDSource level that are handed off to the source plug-in for processing have ctx (RAW) as the first argument. Before calling any of these methods for the first time, the client should allocate the ctx structure, initialize it to NULL, and invoke the openSource( ) method. At this point, the source plug-in can initialize the context for this client. When processing is complete, the client should invoke the closeSource( ) method.

Methods invoked from a source plug-in call have the first argument as ctx (RAW).

Note: In the current release, none of the plug-ins provided by Oracle and not all source or format plug-ins will use the ctx argument, but if you code as previously described, your application should work with current or future source or format plug-ins.

You should use any of the individual set methods to set the attribute value for an object for formats not natively supported; otherwise, for formats natively supported, use the setProperties( ) method to populate the attributes of the object or write a format plug-in.

ORDImage Object Type

Oracle *inter*Media describes the ORDImage object type, which supports the storage, management, and manipulation of image data and the ORDImageSignature object type, which supports content-based retrieval (image matching).

This object type is defined as follows:

```
CREATE OR REPLACE TYPE ORDImage
AS OBJECT
(
  -------------------
  -- TYPE ATTRIBUTES
  -------------------
  source              ORDSource,
  height              INTEGER,
  width               INTEGER,
  contentLength       INTEGER,
  fileFormat          VARCHAR2(4000),
  contentFormat       VARCHAR2(4000),
  compressionFormat   VARCHAR2(4000),
  mimeType            VARCHAR2(4000),

---------------------
-- METHOD DECLARATION
---------------------
-- CONSTRUCTORS
--
STATIC FUNCTION init( ) RETURN ORDImage,
STATIC FUNCTION init(srcType     IN VARCHAR2,
                     srcLocation IN VARCHAR2,
                     srcName     IN VARCHAR2) RETURN ORDImage,

  -- Methods associated with copy operations
  MEMBER PROCEDURE copy(dest IN OUT ORDImage),

  -- Methods associated with image process operations
  MEMBER PROCEDURE process(command IN VARCHAR2),

  MEMBER PROCEDURE processCopy(command IN     VARCHAR2,
                               dest    IN OUT ORDImage),

  -- Methods associated with image property set and check operations
```

```
  MEMBER PROCEDURE setProperties,

  MEMBER PROCEDURE setProperties(description IN VARCHAR2),

  MEMBER FUNCTION checkProperties RETURN BOOLEAN,

  -- Methods associated with image attributes accessors
  MEMBER FUNCTION getHeight RETURN INTEGER,
  PRAGMA RESTRICT_REFERENCES(getHeight, WNDS, WNPS, RNDS, RNPS),

  MEMBER FUNCTION getWidth RETURN INTEGER,
  PRAGMA RESTRICT_REFERENCES(getWidth, WNDS, WNPS, RNDS, RNPS),

  MEMBER FUNCTION getContentLength RETURN INTEGER,
  PRAGMA RESTRICT_REFERENCES(getContentLength, WNDS, WNPS, RNDS, RNPS),

  MEMBER FUNCTION getFileFormat RETURN VARCHAR2,
  PRAGMA RESTRICT_REFERENCES(getFileFormat, WNDS, WNPS, RNDS, RNPS),

  MEMBER FUNCTION getContentFormat RETURN VARCHAR2,
  PRAGMA RESTRICT_REFERENCES(getContentFormat, WNDS, WNPS, RNDS, RNPS),

  MEMBER FUNCTION getCompressionFormat RETURN VARCHAR2,
  PRAGMA RESTRICT_REFERENCES(getCompressionFormat, WNDS, WNPS, RNDS, RNPS),

  -- Methods associated with the local attribute
  MEMBER PROCEDURE  setLocal,
  MEMBER PROCEDURE  clearLocal,
  MEMBER FUNCTION  isLocal RETURN BOOLEAN,
  PRAGMA RESTRICT_REFERENCES(isLocal, WNDS, WNPS, RNDS, RNPS),

  -- Methods associated with the date attribute
  MEMBER FUNCTION  getUpdateTime RETURN DATE,
  PRAGMA RESTRICT_REFERENCES(getUpdateTime, WNDS, WNPS, RNDS, RNPS),
  MEMBER PROCEDURE setUpdateTime(current_time DATE),

  -- Methods associated with the mimeType attribute
  MEMBER FUNCTION  getMimeType RETURN VARCHAR2,
  PRAGMA RESTRICT_REFERENCES(getMimeType, WNDS, WNPS, RNDS, RNPS),
  MEMBER PROCEDURE setMimeType(mime IN VARCHAR2),

  -- Methods associated with the source attribute
MEMBER FUNCTION processSourceCommand(
                                    ctx       IN OUT RAW,
                                    cmd       IN VARCHAR2,
```

```
                                    arguments IN VARCHAR2,
                                    result    OUT RAW)
              RETURN RAW,
  MEMBER FUNCTION  getContent RETURN BLOB,
  PRAGMA RESTRICT_REFERENCES(getContent, WNDS, WNPS, RNDS, RNPS),

  MEMBER FUNCTION getBFILE RETURN BFILE,
  PRAGMA RESTRICT_REFERENCES(getBFILE, WNDS, WNPS, RNDS, RNPS),

  MEMBER PROCEDURE deleteContent,

  MEMBER PROCEDURE setSource(source_type     IN VARCHAR2,
                             source_location IN VARCHAR2,
                             source_name     IN VARCHAR2),
  MEMBER FUNCTION  getSource RETURN VARCHAR2,
  PRAGMA RESTRICT_REFERENCES(getSource, WNDS, WNPS, RNDS, RNPS),

  MEMBER FUNCTION  getSourceType RETURN VARCHAR2,
  PRAGMA RESTRICT_REFERENCES(getSourceType, WNDS, WNPS, RNDS, RNPS),

  MEMBER FUNCTION  getSourceLocation RETURN VARCHAR2,
  PRAGMA RESTRICT_REFERENCES(getSourceLocation, WNDS, WNPS, RNDS, RNPS),

  MEMBER FUNCTION  getSourceName RETURN VARCHAR2,
  PRAGMA RESTRICT_REFERENCES(getSourceName, WNDS, WNPS, RNDS, RNPS),

  MEMBER PROCEDURE import(ctx IN OUT RAW),
  MEMBER PROCEDURE importFrom(ctx         IN OUT RAW,
                         source_type     IN VARCHAR2,
                         source_location IN VARCHAR2,
                         source_name     IN VARCHAR2),
  MEMBER PROCEDURE export(ctx             IN OUT RAW,
                         source_type     IN VARCHAR2,
                         source_location IN VARCHAR2,
                         source_name     IN VARCHAR2),

-- Methods associated with file operations on the source
MEMBER FUNCTION openSource(userArg IN RAW, ctx OUT RAW) RETURN INTEGER,
MEMBER FUNCTION closeSource(ctx IN OUT RAW) RETURN INTEGER,
MEMBER FUNCTION trimSource(ctx     IN OUT RAW,
                           newlen  IN INTEGER) RETURN INTEGER,
MEMBER PROCEDURE readFromSource(
                                ctx      IN OUT RAW,
                                startPos IN INTEGER,
                                numBytes IN OUT INTEGER,
```

```
                                buffer   OUT RAW),
MEMBER PROCEDURE writeToSource(
                               ctx      IN OUT RAW,
                               startPos IN INTEGER,
                               numBytes IN OUT INTEGER,
                               buffer   IN RAW),
);
```

where:

- source: the source of the stored image data.
- height: the height of the image in pixels.
- width: the width of the image in pixels.
- contentLength: the size of the image file on disk, in bytes.
- fileFormat: the file type or format in which the image data is stored (TIFF, JIFF, and so forth).
- contentFormat: the type of image (monochrome and so forth).
- compressionFormat: the compression algorithm used on the image data.
- mimeType: the MIME type information.

ORDImage Constructors

This section describes the ORDImage constructor functions, which are the following:

init( ) for ORDImage

Format

init( ) RETURN ORDImage;

Description

Initializes instances of the ORDImage object type.

Parameters

None.

Pragmas

None.

Exceptions

None.

Usage Notes

This constructor is a static method that initializes all the ORDImage attributes to NULL with the following exceptions:

- source.updateTime is set to SYSDATE
- source.local is set to 1 (local)
- source.localData is set to empty_blob

You should begin using the init( ) method as soon as possible to allow you to more easily initialize the ORDImage object type, especially if the ORDImage type evolves and attributes are added in a future release. INSERT statements left unchanged using the default constructor (which initializes each object attribute), will fail under these circumstances.

Examples

Initialize the ORDImage object attributes:

```
BEGIN
 INSERT INTO pm.online_media (product_id, product_photo)
 VALUES (3501, ORDSYS.ORDImage.init());
 COMMIT;
```

```
END;
/
```

init(srcType,srcLocation,srcName) for ORDImage

Format

```
init(srcType     IN VARCHAR2,
   srcLocation IN VARCHAR2,
   srcName     IN VARCHAR2)
 RETURN ORDImage;
```

Description

Initializes instances of the ORDImage object type.

Parameters

srcType
The source type of the image data.

srcLocation
The source location of the image data.

srcName
The source name of the image data.

Pragmas

None.

Exceptions

None.

Usage Notes

This constructor is a static method that initializes all the ORDImage attributes to NULL with the following exceptions:

- source.updateTime is set to SYSDATE
- source.local is set to 0
- source.localData is set to empty_blob
- source.srcType is set to the input value

- source.srcLocation is set to the input value
- source.srcName is set to the input value

You should begin using the init( ) method as soon as possible to allow you to more easily initialize the ORDImage object type, especially if the ORDImage type evolves and attributes are added in a future release. INSERT statements left unchanged using the default constructor (which initializes each object attribute), will fail under these circumstances.

Examples

Initialize the ORDImage object attributes:

```
BEGIN
 INSERT INTO pm.online_media (product_id, product_photo)
   VALUES (3515, ORDSYS.ORDImage.init('FILE', 'FILE_DIR','speaker.jpg'));
 COMMIT;
END;
/
```

ORDImage Methods

This section presents reference information on the *inter*Media methods used specifically for image data manipulation.

Chapter 3 presents reference information on the *inter*Media methods that are common to ORDAudio, ORDDoc, ORDImage, and ORDVideo. Use the methods presented in both chapters to get and set attributes, perform processing operations, and perform metadata extractions.

The following methods are presented in this section:

checkProperties( )

Format

checkProperties( ) RETURN BOOLEAN;

Description

Verifies that the properties stored in attributes of the image object match the properties of the image. This method should not be used for foreign images (those formats not natively supported by *inter*Media).

Parameters

None.

Usage Notes

Use this method to verify that the image attributes match the actual image.

Pragmas

None.

Exceptions

None.

Examples

Check the image attributes:

```
DECLARE
 image ORDSYS.ORDImage;
 properties_match BOOLEAN;
BEGIN
 SELECT p.product_photo INTO image FROM pm.online_media p
  WHERE p.product_id = 3515;
 -- check that properties match the image
 properties_match := image.checkProperties();
 IF properties_match THEN
  DBMS_OUTPUT.PUT_LINE('Check Properties succeeded');
 ELSE
  DBMS_OUTPUT.PUT_LINE('Check Properties failed');
 END IF;
```

```
COMMIT;
END;
/
```

copy( )

Format

copy(dest IN OUT ORDImage);

Description

Copies an image without changing it.

Parameters

dest
The destination of the new image.

Usage Notes

This method copies the image data, as is, including all source and image attributes, into the supplied local destination image.

If the data is stored locally in the source, then calling this method copies the BLOB to the destination source.localData attribute.

Calling this method copies the external source information to the external source information of the new image, whether or not source data is stored locally.

Calling this method implicitly calls the setUpdateTime( ) method on the destination object to update its timestamp information.

Pragmas

None.

Exceptions

None.

Examples

Create a copy of an image:

```
DECLARE
 image_1 ORDSYS.ORDImage;
 image_2 ORDSYS.ORDImage;
BEGIN
 -- Initialize a new ORDImage object where the copy will be stored:
```

```
  INSERT INTO pm.online_media (product_id, product_photo)
  VALUES (3091, ORDSYS.ORDImage.init());
 -- Select the source object into image_1:
 SELECT product_photo INTO image_1 FROM pm.online_media
   WHERE product_id = 3515;
 -- Select the target object into image_2:
 SELECT product_photo INTO image_2 FROM pm.online_media
   WHERE product_id = 3091 FOR UPDATE;
 -- Copy the data from image_1 to image_2:
 image_1.copy(image_2);
 UPDATE pm.online_media SET product_photo = image_2
  WHERE product_id = 3091;
 COMMIT;
END;
/
```

getCompressionFormat( )

Format

getCompressionFormat( ) RETURN VARCHAR2;

Description

Returns the value of the compressionFormat attribute of the image object.

Parameters

None.

Usage Notes

Use this method rather than accessing the compressionFormat attribute directly to protect yourself from potential changes to the internal representation of the ORDImage object.

Pragmas

PRAGMA RESTRICT_REFERENCES(getCompressionFormat, WNDS, WNPS, RNDS, RNPS)

Exceptions

None.

Examples

Get the compression type of an image:

```
DECLARE
 image ORDSYS.ORDImage;
 compression_format VARCHAR2(4000);
BEGIN
 SELECT p.product_photo INTO image FROM pm.online_media p
  WHERE p.product_id = 3515;
 -- Get the image compression format:
 compression_format := image.getCompressionFormat();
 DBMS_OUTPUT.PUT_LINE('Compression format is ' || compression_format);
 COMMIT;
END;
/
```

getContentFormat( )

Format

getContentFormat( ) RETURN VARCHAR2;

Description

Returns the value of the contentFormat attribute of the image object.

Parameters

None.

Usage Notes

Use this method rather than accessing the contentFormat attribute directly to protect yourself from potential changes to the internal representation of the ORDImage object.

Pragmas

PRAGMA RESTRICT_REFERENCES(getContentFormat, WNDS, WNPS, RNDS, RNPS)

Exceptions

None.

Examples

Get the content type of an image:

```
DECLARE
 image ORDSYS.ORDImage;
 content_format VARCHAR2(4000);
BEGIN
 SELECT p.product_photo INTO image FROM pm.online_media p
  WHERE p.product_id = 3515;
 -- Get the image content format:
 content_format := image.getContentFormat();
 DBMS_OUTPUT.PUT_LINE('Content format is ' || content_format);
 COMMIT;
END;
/
```

getContentLength( )

Format

getContentLength( ) RETURN INTEGER;

Description

Returns the value of the contentLength attribute of the image object.

Parameters

None.

Usage Notes

Use this method rather than accessing the contentLength attribute directly to protect from potential future changes to the internal representation of the ORDImage object.

Pragmas

PRAGMA RESTRICT_REFERENCES(getContentLength, WNDS, WNPS, RNDS, RNPS)

Exceptions

None.

Examples

Get the content length of an image:

```
DECLARE
 image ORDSYS.ORDImage;
 content_length INTEGER;
BEGIN
 SELECT p.product_photo INTO image FROM pm.online_media p
  WHERE p.product_id = 3515;
 -- Get the image size:
 content_length := image.getContentLength();
 DBMS_OUTPUT.PUT_LINE('Content length is ' || content_length);
 COMMIT;
END;
/
```

getFileFormat( )

Format

getFileFormat( ) RETURN VARCHAR2;

Description

Returns the value of the fileFormat attribute of the image object.

Parameters

None.

Usage Notes

Use this method rather than accessing the fileFormat attribute directly to protect yourself from potential changes to the internal representation of the ORDImage object.

Pragmas

PRAGMA RESTRICT_REFERENCES(getFileFormat, WNDS, WNPS, RNDS, RNPS)

Exceptions

None.

Examples

Get the file type of an image:

```
DECLARE
 image ORDSYS.ORDImage;
 file_format VARCHAR2(4000);
BEGIN
 SELECT p.product_photo INTO image FROM pm.online_media p
  WHERE p.product_id = 3515;
 -- Get the image file format:
 file_format := Image.getFileFormat();
 DBMS_OUTPUT.PUT_LINE('File format is ' || file_format);
COMMIT;
END;
/
```

getHeight( )

Format

getHeight( ) RETURN INTEGER;

Description

Returns the value of the height attribute of the image object.

Parameters

None.

Usage Notes

Use this method rather than accessing the height attribute directly to protect yourself from potential changes to the internal representation of the ORDImage object.

Pragmas

PRAGMA RESTRICT_REFERENCES(getHeight, WNDS, WNPS, RNDS, RNPS)

Exceptions

None.

Examples

Get the height of an image:

```
DECLARE
 image ORDSYS.ORDImage;
 height INTEGER;
BEGIN
 SELECT p.product_photo INTO image FROM pm.online_media p
  WHERE p.product_id = 3515;
 -- Get the image height:
 height := image.getHeight();
 DBMS_OUTPUT.PUT_LINE('Height is ' || height);
 COMMIT;
END;
/
```

getWidth( )

Format

getWidth( ) RETURN INTEGER;

Description

Returns the value of the width attribute of the image object

Parameters

None.

Usage Notes

Use this method rather than accessing the width attribute directly to protect yourself from potential changes to the internal representation of the ORDImage object.

Pragmas

PRAGMA RESTRICT_REFERENCES(getWidth, WNDS, WNPS, RNDS, RNPS)

Exceptions

None.

Examples

Get the width of an image:

```
DECLARE
 image ORDSYS.ORDImage;
 width INTEGER;
BEGIN
 SELECT p.product_photo INTO image FROM pm.online_media p
  WHERE p.product_id = 3515;
 -- Get the image width:
 width := image.getWidth();
 DBMS_OUTPUT.PUT_LINE('Width is ' || width);
 COMMIT;
END;
/
```

import( )

Format

```
import(ctx IN OUT RAW);
```

Description

Transfers image data from an external image data source to the localData attribute (of the embedded ORDSource object) within the database.

Parameters

ctx
The source plug-in context information. This should be allocated and initialized to NULL. If you are using a user-defined source plug-in, you should call the openSource( ) method. See the introduction to this chapter for more information.

Usage Notes

Use the setSource( ) method to set the source.srcType, source.srcLocation, and source.srcName attributes (of the embedded ORDSource object) for the external image data source prior to calling the import( ) method.

You must ensure that the directory exists or is created before you use this method for a source.srcType attribute with a value of "file".

After importing data from an external image data source to a local source (within Oracle Database), the source information remains unchanged (that is, pointing to the source from where the data was imported).

Invoking this method implicitly calls the setUpdateTime( ) and setLocal( ) methods.

If the file format of the imported image is not previously set to a string beginning with "OTHER", the setProperties( ) method is also called. Set the file format to a string preceded by "OTHER" for foreign image formats; calling the setProperties( ) method for foreign images does this for you.

This method is invoked at the ORDSource level, which uses the PL/SQL UTL_HTTP package to import media data from an HTTP data source. You can use environment variables to specify the proxy behavior of the UTL_HTTP package. For example, on UNIX, setting the environment variable http_proxy to a URL specifies that the UTL_HTTP package will use that URL as the proxy server for HTTP

requests. Setting the no_proxy environment variable to a domain name specifies that the HTTP proxy server will not be used for URLs in the specified domain.

See *PL/SQL Packages and Types Reference* for more information about the UTL_HTTP PL/SQL package.

Pragmas

None.

Exceptions

ORDSourceExceptions.INCOMPLETE_SOURCE_INFORMATION

This exception is raised if you call the import( ) method and the value of the source.srcType attribute is NULL.

ORDSourceExceptions.METHOD_NOT_SUPPORTED

This exception is raised if you call the import( ) method and this method is not supported by the source plug-in being used.

See Appendix F for more information about these exceptions.

Examples

Import image data from an external image data source into the local source:

```
DECLARE
 obj ORDSYS.ORDImage;
 ctx RAW(64) :=NULL;
BEGIN
 SELECT p.product_photo INTO obj FROM pm.online_media p
  WHERE p.product_id  = 3515 FOR UPDATE;
 DBMS_OUTPUT.PUT_LINE('getting source');
 DBMS_OUTPUT.PUT_LINE('-------------------------');
 -- Get source information
 DBMS_OUTPUT.PUT_LINE(obj.getSource());
 -- Import data
 obj.import(ctx);
 -- Check size
 DBMS_OUTPUT.PUT_LINE('Length is ' || obj.getContentLength);
 UPDATE pm.online_media p SET p.product_photo = obj WHERE p.product_id = 3515;
 COMMIT;
END;
/
```

importFrom( )

Format

```
importFrom(ctx                IN OUT RAW,
           source_type        IN VARCHAR2,
           source_location    IN VARCHAR2,
           source_name        IN VARCHAR2);
```

Description

Transfers image data from the specified external image data source to the source.localData attribute (of the embedded ORDSource object) within the database.

Parameters

ctx
The source plug-in context information. This should be allocated and initialized to NULL. If you are using a user-defined source plug-in, you should call the openSource( ) method. See the introduction to this chapter for more information.

source_type
The type of the source image data.

source_location
The location from which the source image data is to be imported.

source_name
The name of the source image data.

Usage Notes

This method is similar to the import( ) method except the source information is specified as parameters to the method instead of separately.

You must ensure that the directory exists or is created before you use this method with a source_type parameter value of "file".

After importing data from an external image data source to a local source (within Oracle Database), the source information (that is, pointing to the source from where the data was imported) is set to the input values.

Invoking this method implicitly calls the setUpdateTime( ) and setLocal( ) methods.

If the file format of the imported image is not previously set to a string beginning with "OTHER", the setProperties( ) method is also called. Set the file format to a string preceded by "OTHER" for foreign image formats; calling the setProperties( ) for foreign images method does this for you.

This method is invoked at the ORDSource level, which uses the PL/SQL UTL_HTTP package to import media data from an HTTP data source. You can use environment variables to specify the proxy behavior of the UTL_HTTP package. For example, on UNIX, setting the environment variable http_proxy to a URL specifies that the UTL_HTTP package will use that URL as the proxy server for HTTP requests. Setting the no_proxy environment variable to a domain name specifies that the HTTP proxy server will not be used for URLs in the specified domain.

See *PL/SQL Packages and Types Reference* for more information about the UTL_HTTP PL/SQL package.

Pragmas

None.

Exceptions

ORDSourceExceptions.METHOD_NOT_SUPPORTED

This exception is raised if you call the importFrom( ) method and this method is not supported by the source plug-in being used.

ORDSourceExceptions.SOURCE_PLUGIN_EXCEPTION

This exception is raised if you call the importFrom( ) method and the source plug-in raises an exception.

See Appendix F for more information about these exceptions.

Examples

Import image data from the specified external data source into the local source:

```
DECLARE
 obj ORDSYS.ORDImage;
 ctx RAW(64) :=NULL;
BEGIN
 SELECT p.product_photo INTO obj FROM pm.online_media p
  WHERE p.product_id  = 3501 FOR UPDATE;
 -- set source to a file
```

```
 -- import data
 obj.importFrom(ctx,'file','FILE_DIR','speaker.jpg');
 -- check size
 DBMS_OUTPUT.PUT_LINE('Length is ' || obj.getContentLength);
 DBMS_OUTPUT.PUT_LINE('Source is ' || obj.getSource());
 UPDATE pm.online_media p SET p.product_photo = obj WHERE p.product_id = 3501;
 COMMIT;
END;
/
```

process( )

Format

process(command IN VARCHAR2);

Description

Performs one or more image processing operations on a BLOB, writing the image back onto itself.

Parameters

command
A list of image processing operations to perform on the image.

Usage Notes

There is no implicit import( ) or importFrom( ) call performed when you call this method; if data is external, you must first call the import( ) or importFrom( ) method to make the data local before you can process it.

Implicit setProperties( ), setUpdateTime( ), and setMimeType( ) methods are done after the process( ) method is called.

See Appendix D for more information on process( ) method operators.

You can change one or more of the image attributes shown in Table 6–1. Table 6–2 shows additional changes that can be made only to raw pixel and foreign images.

Table 6–1 Image Processing Operators

Operator Name	Usage	Values
compressionFormat	Forces output to the specified compression format if it is supported by the output file format.	JPEG, SUNRLE, BMPRLE, TARGARLE, LZW, LZWHDIFF, FAX3, FAX4, HUFFMAN3, PACKBITS, GIFLZW, ASCII, RAW, DEFLATE, NONE
compressionQuality	Determines the quality of lossy compression; for use with JPEG only.	MAXCOMPRATIO, MAXINTEGRITY, LOWCOMP, MEDCOMP, HIGHCOMP, or an integer value between 0 and 100.
contentFormat	Determines the format of the image content.	See Section D.2.2 for values.

Table 6–1 (Cont.) Image Processing Operators

Operator Name	Usage	Values
contrast	Adjusts image contrast. See Section D.3.1 for more information.	nonnegative FLOAT[1], nonnegative FLOAT FLOAT FLOAT[2], nonnegative FLOAT FLOAT[3], nonnegative FLOAT FLOAT FLOAT FLOAT FLOAT FLOAT[4]
cut	Defines a window to cut or crop (origin.x origin.y width height); first pixel in x or y is 0 (zero); must define a window inside image.	nonnegative INTEGER INTEGER INTEGER INTEGER maximum value is 2147483648
fileFormat	Forces the output to specified file format.	BMPF, CALS, GIFF, JFIF, PBMF, PGMF, PICT, PNGF, PNMF, PPMF, RASF, RPIX, TGAF, TIFF, WBMP
fixedScale	Scales an image to a specified size in pixels (width, height); may not be combined with other scale operators.	positive INTEGER INTEGER
flip	Places the scanlines of an image in inverse order -- swapped top to bottom.	No arguments
gamma	Adjusts gamma (brightness) of an image. See Section D.3.4 for more information.	positive FLOAT[5] positive FLOAT FLOAT FLOAT[6]
maxScale	Scales an image to a specified size in pixels (width, height), while maintaining the aspect ratio; may not be combined with other scale operators.	positive INTEGER INTEGER
mirror	Places columns of an image in reverse order -- swapped left to right.	No arguments
page	Selects a page from a multipage file; for use with TIFF only; first page is 0 (zero).	nonnegative INTEGER

Table 6–1 (Cont.) Image Processing Operators

Operator Name	Usage	Values
quantize	Specifies how image quantization is to be performed when reducing image bit depth. See Section D.3.7 for more information.	ERRORDIFFUSION (default), ORDEREDDITHER, THRESHOLD, MEDIANCUT
rotate	Rotates an image within the image plane by the angle specified. See Section D.3.8 for more information.	FLOAT
scale	Uniformly scales an image by a given factor (for example, 0.5 or 2.0); may not be combined with other scale operators.	positive FLOAT
tiled	Forces output image to be tiled; for use with TIFF only.	No arguments
xScale	Scales an image on the X-axis by a given factor (default is 1); image is non-uniformly scaled; may be combined only with the yScale operator; may not be combined with any other scale operators.	positive FLOAT
yScale	Scales the image on theY-axis scale by a given factor (default is 1); non-uniformly scales image; may only be combined with the xScale operator; may not be combined with any other scale operators.	positive FLOAT

[1] Specifies the percent contrast enhancement to be applied to all bands (GRAY or RGB)

[2] Specifies the percent contrast enhancement to be applied to each band (RGB only)

[3] Specifies the bounds for contrast enhancement to be applied to all bands (GRAY or RGB)

[4] Specifies the bounds for contrast enhancement to be applied to each band (RGB only)
[5] Specifies a gamma value to be applied to all bands (GRAY or RGB)
[6] Specifies separate gamma values to be applied to each band (RGB only)

Note: When specifying values that include floating-point numbers, you must use double quotation marks (" ") around the value. If you do not, the wrong values may be passed and you will get incorrect results.

Table 6–2 Additional Image Processing Operators for Raw Pixel and Foreign Images

Operator Name	Usage	Values
channelOrder	Indicates the relative position of the red, green, and blue channels (bands) within the image; changes order of output channels. Only for RPIX.	RGB (default), RBG, GRB, GBR, BRG, BGR
inputChannels	For multiband images, specifies either one (grayscale) or three integers indicating which channels to assign to red (first), green (second), and blue (third). Note that this operator affects the source image, not the destination; RPIX only.	positive INTEGER,[1] positive INTEGER INTEGER INTEGER[2]
pixelOrder	Forces pixel direction. If NORMAL, then the leftmost pixel appears first in the image. RPIX only.	NORMAL (default), REVERSE
scanlineOrder	Forces scanline direction. If NORMAL, then the top scanline appears first in the image. RPIX and BMPF only.	NORMAL (default), INVERSE

[1] Specifies that a single band is to be selected from the input image and that band will be used to create a grayscale output image
[2] Specifies that three bands are to be selected from the input image and those bands will specify the red, green, and blue bands of an RGB output image

Pragmas

None.

Exceptions

ORDImageExceptions.DATA_NOT_LOCAL

This exception is raised if you call the process( ) method and the data is not local (the source.local attribute is 0).

Examples

Example 1: Change the file format of image1 to GIFF:

```
DECLARE
 obj ORDSYS.ORDImage;
BEGIN
 SELECT product_photo INTO obj FROM pm.online_media
  WHERE product_id = 3515 FOR UPDATE;
 obj.process('fileFormat=GIFF');
 -- Update
 UPDATE pm.online_media SET product_photo = obj WHERE product_id = 3515;
 -- Roll back to keep original format of image:
 ROLLBACK;
 EXCEPTION
  WHEN ORDSYS.ORDImageExceptions.DATA_NOT_LOCAL THEN
   DBMS_OUTPUT.PUT_LINE('Data is not local');
END;
/
```

Example 2: Change image1 to use a compression format of JPEG with MAXCOMPRATIO and double the length of the image along the X-axis:

```
DECLARE
 obj ORDSYS.ORDImage;
BEGIN
 SELECT product_photo INTO obj FROM pm.online_media
  WHERE product_id = 3515 FOR UPDATE;
  obj.process(
       'compressionFormat=JPEG,compressionQuality=MAXCOMPRATIO, xScale="2.0"');
 -- Update:
 UPDATE pm.online_media SET product_photo = obj WHERE product_id = 3515;
 -- Roll back to keep original format of image:
 ROLLBACK;
 EXCEPTION
  WHEN ORDSYS.ORDImageExceptions.DATA_NOT_LOCAL THEN
   DBMS_OUTPUT.PUT_LINE('Data is not local');
END;
/
```

Note that changing the length on only one axis (for example, xScale=2.0) does not affect the length on the other axis, and would result in image distortion. Also, only the xScale and yScale parameters can be combined in a single scale operation. Any other combinations of scale operators result in an error.

Example 3: Create a thumbnail image:

The maxScale and fixedScale operators are especially useful for creating thumbnail images from various-sized originals. The following example creates, at most, a 32-by-32 pixel thumbnail image, preserving the original aspect ratio:

```
DECLARE
 obj ORDSYS.ORDImage;
BEGIN
 SELECT product_photo INTO obj FROM pm.online_media
  WHERE product_id = 3515 FOR UPDATE;
 obj.process('maxScale=32 32');
 UPDATE pm.online_media p SET product_thumbnail = obj
  WHERE product_id = 3515;
 COMMIT;
 EXCEPTION
  WHEN ORDSYS.ORDImageExceptions.DATA_NOT_LOCAL THEN
   DBMS_OUTPUT.PUT_LINE('Data is not local');
END;
/
```

Example 4: Change the format to TIFF and the content format to 8BIT, BIP pixel layout, LUT interpretation, and RGB color space:

```
DECLARE
 obj ORDSYS.ORDImage;
BEGIN
 SELECT product_photo INTO obj FROM pm.online_media
  WHERE product_id = 3515 FOR UPDATE;
 obj.process('fileFormat=TIFF, contentFormat=8BITBIPLUTRGB');
 UPDATE pm.online_media SET product_photo = obj WHERE product_id = 3515;
 -- Roll back to keep original format of image:
 ROLLBACK;
 EXCEPTION
  WHEN ORDSYS.ORDImageExceptions.DATA_NOT_LOCAL THEN
   DBMS_OUTPUT.PUT_LINE('Data is not local');
END;
/
```

processCopy( )

Format

```
processCopy(command IN     VARCHAR2,
            dest           IN OUT ORDImage);
```

Description

Copies an image stored internally or externally to another image stored internally in the source.LocalData attribute (of the embedded ORDSource object) and performs one or more image processing operations on the copy.

Parameters

command
A list of image processing changes to make for the image in the new copy.

dest
The destination of the new image.

Usage Notes

See Table 6–1, " Image Processing Operators" and Table 6–2, " Additional Image Processing Operators for Raw Pixel and Foreign Images".

You cannot specify the same ORDImage as both the source and destination.

Calling this method processes the image into the destination BLOB from any source (local or external).

Implicit setProperties( ), setUpdateTime( ), and setMimeType( ) methods are applied on the destination image after the processCopy( ) method is called.

See Appendix D for more information on processCopy( ) operators.

Pragmas

None.

Exceptions

ORDImageExceptions.NULL_DESTINATION

This exception is raised if you call the processCopy( ) method and the destination image is NULL.

ORDImageExceptions.DATA_NOT_LOCAL

This exception is raised if you call the processCopy( ) method and the destination image source.local attribute value is 0 or the destination source.localData attribute is not initialized.

ORDImageExceptions.NULL_LOCAL_DATA

This exception is raised if you call the processCopy( ) method and the destination image source.localData attribute value is NULL.

This exception is raised if you call the processCopy( ) method and the source image source.local attribute value is 1 or NULL and the source.localData attribute value is NULL.

Examples

Generate a thumbnail image from a source image:

```
DECLARE
 obj_1 ORDSYS.ORDImage;
 obj_2 ORDSYS.ORDImage;

BEGIN
 SELECT product_photo, product_thumbnail INTO obj_1, obj_2
  FROM pm.online_media
  WHERE product_id = 3515 FOR UPDATE;
  obj_1.processCopy('maxScale=32 32', obj_2);
 UPDATE pm.online_media SET product_thumbnail = obj_2
  WHERE product_id=3515;
 COMMIT;
EXCEPTION
  WHEN ORDSYS.ORDImageExceptions.NULL_DESTINATION THEN
   DBMS_OUTPUT.PUT_LINE('The destination is null');
  WHEN ORDSYS.ORDImageExceptions.DATA_NOT_LOCAL THEN
   DBMS_OUTPUT.PUT_LINE('Data is not local');
  WHEN ORDSYS.ORDImageExceptions.NULL_LOCAL_DATA THEN
   DBMS_OUTPUT.PUT_LINE('dest.source.localData attribute is null');
COMMIT;
END;
/
```

setProperties( )

Format

```
setProperties( );
```

Description

Reads the image data to get the values of the object attributes, then stores them into the appropriate attribute fields. The image data can be stored in the database the source.localData attribute, or externally in a BFILE or URL. If the data is stored externally in anything other than a BFILE, the data is read into a temporary BLOB in order to determine the image characteristics.

This method should not be called for foreign images. Use the setProperties( ) for foreign images method instead.

Parameters

None.

Usage Notes

After you have copied, stored, or processed a native format image, call this method to set the current characteristics of the new content, except when this method is called implicitly.

This method sets the following information about an image:

- Height in pixels
- Width in pixels
- Data size of the image on disk, in bytes
- File type (TIFF, JFIF, and so forth)
- Image type (monochrome and so forth)
- Compression type (JPEG, LZW, and so forth)
- MIME type (generated based on file format)

Calling this method implicitly calls the setUpdateTime( ) and the setMimeType( ) methods.

Pragmas

None.

Exceptions

ORDImageExceptions.NULL_LOCAL_DATA

This exception is raised if you call the setProperties( ) method and the source.local attribute value is 1 or NULL and the source.localData attribute value is NULL.

Examples

Select the image, and then set the attributes using the setProperties( ) method:

```
DECLARE
 image ORDSYS.ORDImage;
BEGIN
 SELECT p.product_photo INTO image FROM pm.online_media p
  WHERE p.product_id = 3515 FOR UPDATE;
 -- set property attributes for the image data
 image.setProperties();
 DBMS_OUTPUT.PUT_LINE('image width = ' || image.getWidth());
 DBMS_OUTPUT.PUT_LINE('image height = ' || image.getHeight());
 DBMS_OUTPUT.PUT_LINE('image size = ' || image.getContentLength());
 DBMS_OUTPUT.PUT_LINE('image file type = ' || image.getFileFormat());
 DBMS_OUTPUT.PUT_LINE('image type = ' || image.getContentFormat());
 DBMS_OUTPUT.PUT_LINE('image compression = ' || image.getCompressionFormat());
 DBMS_OUTPUT.PUT_LINE('image mime type = ' || image.getMimeType());
 UPDATE pm.online_media p SET p.product_photo = image
   WHERE p.product_id = 3515;
 COMMIT;
END;
/
```

setProperties( ) for foreign images

Format

```
setProperties(description IN VARCHAR2);
```

Description

Lets you write the characteristics of certain foreign images whose format is not natively understood by *inter*Media into the appropriate attribute fields.

Parameters

description
The image characteristics to set for the foreign image.

Usage Notes

> **Note:** Once you have set the properties for a foreign image, it is up to you to keep the properties consistent. If *inter*Media detects an unknown file format, it will not implicitly set the properties.

See Appendix E for information on when to use foreign image support. Only some formats that are not natively understood can be described using this setProperties( ) method.

After you have copied, stored, or processed a foreign image, call this method to set the characteristics of the new image content. Unlike the native image types described in Appendix B, foreign images either do not contain information on how to interpret the bits in the file or, *inter*Media does not understand the information. In this case, you must set the information explicitly.

You can set the image characteristics for foreign images as described in Table 6–3.

Table 6–3 Image Characteristics for Foreign Files

Field	Data Type	Description
CompressionFormat	STRING	Specifies the compression format value. Valid values are: CCITTG3, CCITTG4, or NONE (default).
DataOffset	INTEGER	Specifies an offset (from the beginning of the file to the start of the image data) that *inter*Media does not try to interpret. Set the offset to skip any potential header. The value must be a nonnegative integer less than the LOB length. Default is zero.
DefaultChannelSelection	INTEGER or INTEGER, INTEGER, INTEGER	Specifies which bands in a multiband image will be interpreted as color channels when the image is read or processed. If a single integer is specified, the image will be treated as a grayscale image consisting of the data in the specified band only. If three integers are specified, the image will be treated as an RGB image, using the first specified band as the red channel, the second as the green channel, and the third as the blue channel. The first band in the image is numbered 1. The band specifications must be equal to or lower than the number of bands in the image. For example: ■ Specify "DefaultChannelSelection = 1" to cause the first band of the image to be interpreted as a grayscale image. ■ Specify "DefaultChannelSelection = 4" to cause the fourth band of the image to be interpreted as a grayscale image. ■ Specify "DefaultChannelSelection = 1, 2, 3" to cause the image to be interpreted as RGB using the first three bands of the image as red, green and blue channels. ■ Specify "DefaultChannelSelection = 3, 1, 4" to cause the image to be interpreted as RGB using the third, first, and fourth bands of the image as the red, green and blue channels.
Height	INTEGER	Specifies the height of the image in pixels. Value must be a positive integer. There is no default, thus a value must be specified.
Interleaving	STRING	Specifies the band layout within the image. Valid styles are: ■ BIP (default) Band Interleaved by Pixel ■ BIL Band Interleaved by Line ■ BSQ Band Sequential
NumberOfBands	INTEGER	Specifies the number of color bands in the image with a value that is a positive integer less than 255. Default is 3.
PixelOrder	STRING	Specifies a string to indicate the pixel order. If the string is NORMAL (default), the leftmost pixel appears first in the file. If the string is REVERSE, the rightmost pixel appears first.

Table 6–3 (Cont.) Image Characteristics for Foreign Files

Field	Data Type	Description
ScanlineOrder	STRING	Specifies a string to indicate the scanline order. If the string is NORMAL (default), the top scanline appears first in the file. If the string is INVERSE, then the bottom scanline appears first.
UserString	STRING	Specifies a 4-character descriptive string. If used, the string is stored in the fileFormat attribute, appended to the user string " OTHER:". Default is blank and fileFormat is set to "OTHER".
Width	INTEGER	Specifies the width of the image in pixels. Value must be a positive integer. There is no default, thus a value must be specified.
MimeType	STRING	Specifies a MIME type, such as img/gif.

The values supplied to the setProperties( ) for foreign images method are written to the existing ORDImage data attributes. The fileFormat attribute is set to OTHER and includes the user string, if supplied; for example, OTHER: LANDSAT.

Pragmas

None.

Exceptions

ORDImageExceptions.NULL_PROPERTIES_DESCRIPTION

This exception is raised if you call the setProperties( ) for foreign images method and the description parameter value is NULL.

Examples

Select the foreign image and then set the properties for the image:

```
DECLARE
    obj ORDSYS.ORDImage;
BEGIN
 SELECT p.product_photo INTO obj FROM pm.online_media p
  WHERE p.product_id = 3501 FOR UPDATE;
 -- Set property attributes for the image data:
 obj.setProperties('width=123 height=321 compressionFormat=NONE' ||
                   ' userString= LANDSTAT dataOffset=128' ||
                   ' scanlineOrder=INVERSE pixelOrder=REVERSE' ||
                   ' interleaving=BIL numberOfBands=1' ||
                   ' defaultChannelSelection=1');
 UPDATE pm.online_media SET product_photo = obj
```

```
   WHERE product_id=3501;
 COMMIT;
END;
/
```

ORDImageSignature Object Type

Oracle *inter*Media describes the ORDImageSignature object type, which supports content-based retrieval (image matching).

> **Note:** All *inter*Media features are available with the Standard Edition of Oracle Database except image indexing. The image indexing feature requires bit-mapped indexing, which is available only when you install the Enterprise Edition of Oracle Database.
>
> See the information on using an index to compare signatures in *Oracle interMedia User's Guide* for details on image indexing.

The examples in this chapter use the ONLINE_MEDIA table in the Product Media sample schema. See *Oracle Database Sample Schemas* for information on the sample schemas.

> **Note:** When you are storing or copying ORDImageSignature objects, you must first create an empty ORDImageSignature object in the table by using the ORDImageSignature.init( ) method.

The ORDImageSignature object type supports content-based retrieval, or image matching. This object type is defined as follows:

```
CREATE OR REPLACE TYPE ORDImageSignature
AS OBJECT
(
  -- Signature of the image.  Contains color, texture
  -- and shape information of the image.  It is stored
  -- in a BLOB.

  signature BLOB,

-----------------------
-- METHOD DECLARATION

-- Makes the callout

STATIC FUNCTION init RETURN ORDImageSignature,
```

```
STATIC FUNCTION evaluateScore(sig1    IN ORDImageSignature,
                              sig2    IN ORDImageSignature,
                              weights IN VARCHAR2)
  RETURN FLOAT,

STATIC FUNCTION isSimilar(sig1      IN ORDImageSignature,
                          sig2      IN ORDImageSignature,
                          weights   IN VARCHAR2,
                          threshold IN FLOAT)
  RETURN INTEGER,

MEMBER PROCEDURE generateSignature(image  IN ORDImage)
);
```

where:

- signature: holds the signature of the stored image data.

ORDImageSignature Constructor

This section describes the constructor function.

The *inter*Media constructor function is as follows:

- init( ) for ORDImageSignature on page 6-45

init( ) for ORDImageSignature

Format

init( ) RETURN ORDImageSignature;

Description

Initializes instances of the ORDImageSignature object type.

Parameters

None.

Pragmas

None.

Exceptions

None.

Usage Notes

This constructor is a static method that initializes the ORDImageSignature signature attribute to empty_blob.

You must use the init( ) method to initialize ORDImageSignature objects that are stored in the database.

Examples

Initialize the ORDImageSignature object attribute:

```
BEGIN
 INSERT INTO pm.online_media (product_id, product_photo,
  product_photo_signature)
  VALUES (1910, ORDSYS.ORDImage.init('FILE', 'FILE_DIR','speaker.jpg'),
  ORDSYS.ORDImageSignature.init());
 INSERT INTO pm.online_media (product_id, product_photo,
  product_photo_signature)
  VALUES (1940, ORDSYS.ORDImage.init('FILE', 'FILE_DIR','keyboard.jpg'),
  ORDSYS.ORDImageSignature.init());
 INSERT INTO pm.online_media (product_id, product_photo,
  product_photo_signature)
  VALUES (2402, ORDSYS.ORDImage.init('FILE', 'FILE_DIR','speaker_02.jpg'),
```

```
  ORDSYS.ORDImageSignature.init());
COMMIT;
END;
/
```

ORDImageSignature Methods

This section presents reference information on the *inter*Media methods used for image data manipulation.

The following methods are presented in this section:

evaluateScore( )

Format

```
evaluateScore(
                sig1 IN ORDImageSignature,
                sig2 IN ORDImageSignature,
                weights IN VARCHAR2)
RETURN FLOAT;
```

Description

A static method of the ORDImageSignature object type that evaluates the distance between two input signatures based on the influence of the specified attributes in the weights parameter.

Parameters

sig1
A signature object.

sig2
A signature object.

weights
A string consisting of matching attribute names followed by values between 0.0 and 1.0. The matching attributes refer to the weights assigned by the user to the different attributes that influence the kind of match. Attributes not specified by the user have a default value of 0.0.

The string can have all or some of the attributes. The value associated with an attribute specifies its relative importance in determining the distance between the signatures. An attribute with a value of 0.0 is not considered at all, while an attribute with a value of 1.0 is of the highest importance. The weights supplied are normalized prior to processing such that they add up to 1.0, maintaining the ratios that you supplied. At least one of the attributes must have a value greater than 0.0. The attributes are the following:

- color: A value between 0.0 and 1.0 indicating the importance of the feature color.

- texture: A value between 0.0 and 1.0 indicating the importance of the feature texture.
- shape: A value between 0.0 and 1.0 indicating the importance of the feature shape.
- location: A value between 0.0 and 1.0 indicating the importance of the location of the regions in the image. The location weight string cannot be specified alone, it must be used with another weight string.

Return Value

This function returns a FLOAT value between 0.0 and 100.0, where 0.0 means the images are identical and 100.0 means the images are completely different.

Usage Notes

The ORDImageSignature evaluateScore( ) method operates on two image signatures, not on indexes on database tables. Therefore, this method cannot take advantage of the increased performance that is possible using image matching with image signature indexes on the underlying tables. To use signature images, use the IMGSimilar and IMGScore SQL operators.

The ORDImageSignature objects must either be initialized, inserted into a table, and have a signature generated for them, or be created using a temporary LOB and have a signature generated for them, to successfully use the evaluateScore( ) method to compare the signatures.

Pragmas

None.

Exceptions

None.

Examples

Compare two signatures and evaluate the score between them:

```
DECLARE
 t_image     ORDSYS.ORDImage;
 c_image    ORDSYS.ORDImage;
 image_sig   ORDSYS.ORDImageSignature;
 compare_sig ORDSYS.ORDImageSignature;
 score FLOAT;
```

```
BEGIN
 SELECT p.product_photo, p.product_photo_signature INTO t_image, image_sig
  FROM pm.online_media p
  WHERE p.product_id = 1910 FOR UPDATE;
 -- Generate a signature:
 image_sig.generateSignature(t_image);
 UPDATE pm.online_media p SET p.product_photo_signature = image_sig
  WHERE product_id =1910;
 SELECT p.product_photo, p.product_photo_signature INTO c_image, compare_sig
  FROM pm.online_media p
  WHERE p.product_id = 1940 FOR UPDATE;
 -- Generate a signature:
 compare_sig.generateSignature(c_image);
 UPDATE pm.online_media p SET p.product_photo_signature = compare_sig
  WHERE product_id = 1940;
SELECT p.product_photo, p.product_photo_signature INTO t_image, image_sig
 FROM pm.online_media p
 WHERE p.product_id  = 1910;
SELECT p.product_photo, p.product_photo_signature INTO c_image, compare_sig
 FROM pm.online_media p
 WHERE p.product_id = 1940;
 -- Compare two images for similarity based on image color:
 score:=ORDSYS.ORDImageSignature.evaluateScore(image_sig,
  compare_sig,'color=1.0,texture=0,shape=0,location=0');
 DBMS_OUTPUT.PUT_LINE('Score is ' || score);
END;
/
```

generateSignature( )

Format

generateSignature (image IN ORDImage);

Description

Generates a signature for a given input image that is passed back as the signature object.

Parameters

image
The image object whose signature is to be generated.

Usage Notes

The ORDImageSignature object must either be initialized and inserted into a table or be created using a temporary LOB, to successfully generate a signature for the object.

The minimum image size for which the generateSignature( ) method can be called is 21x21 pixels. The generateSignature( ) method fails and returns the "IMG-00870, Unsupported aspect ratio or image size" error when called for smaller images.

Pragmas

None.

Exceptions

None.

Examples

Generate a signature for an image object stored in the database:

```
DECLARE
 t_image   ORDSYS.ORDImage;
 image_sig ORDSYS.ORDImageSignature;
BEGIN
 SELECT p.product_photo, p.product_photo_signature INTO t_image, image_sig
  FROM pm.online_media p
  WHERE p.product_id = 2402 FOR UPDATE;
```

```
 -- Generate a signature:
 image_sig.generateSignature(t_image);
 UPDATE pm.online_media p SET p.product_photo_signature = image_sig
 WHERE    product_id = 2402;
END;
/
```

isSimilar( )

Format

```
isSimilar(
                sig1 IN ORDImageSignature,
                sig2 IN ORDImageSignature,
                weights IN VARCHAR2,
                threshold IN FLOAT)
RETURN INTEGER;
```

Description

A static method of the ORDImageSignature object type that compares two signatures, and computes the distance between them based on the influence of the specified attributes in the weights parameter and the specified threshold value. If the distance is less than the specified threshold, a value of 1 is returned, otherwise a value of 0 is returned.

Parameters

sig1
A signature object.

sig2
A signature object.

weights
A string consisting of matching attribute names followed by values between 0.0 and 1.0. The matching attributes refer to the weights assigned by the user to the different attributes that influence the kind of match. Attributes not specified by the user have a default value of 0.0.

The string can have all or some of the attributes. The value associated with an attribute specifies its relative importance in determining the distance between the signatures. An attribute with a value of 0.0 is not considered at all, while an attribute with a value of 1.0 is of the highest importance. The weights supplied are normalized prior to processing such that they add up to 1.0, maintaining the ratios that you supplied. At least one of the attributes must have a value greater than 0.0. The attributes are the following:

- color: A value between 0.0 and 1.0 indicating the importance of the feature color.
- texture: A value between 0.0 and 1.0 indicating the importance of the feature texture.
- shape: A value between 0.0 and 1.0 indicating the importance of the feature shape.
- location: A value between 0.0 and 1.0 indicating the importance of the location of the regions in the image. The location weight string cannot be specified alone, it must be used with another weight string.

threshold
The degree of the match that the user desires. For example, if the value is specified as 10, then only those images whose signatures are a distance of 10 or less (score of 10 or less) from the query signature will be returned. The value of the threshold ranges from 0 to 100, which is the range of the distance.

Usage Notes

You can use this method to compare two signatures not stored in the database, or when you must perform a comparison within a PL/SQL construct.

The ORDImageSignature isSimilar( ) method operates on two image signatures, not on indexes on database tables. Therefore, this method cannot take advantage of the increased performance that is possible using image matching with image signature indexes on the underlying tables. To use signature images, use the IMGSimilar and IMGScore SQL operators.

The ORDImageSignature objects must either be initialized, inserted into a table, and have a signature generated for them, or be created using a temporary LOB and have a signature generated for them, to successfully use the isSimilar( ) method to compare the signatures.

Pragmas

None.

Exceptions

None.

Examples

Compare two images based on color. Images are considered similar if a distance of 10 or less is returned:

```
DECLARE
 image_sig1 ORDSYS.ORDImageSignature;
 image_sig2 ORDSYS.ORDImageSignature;
 value     INTEGER;
BEGIN
 SELECT product_photo_signature INTO image_sig1 FROM pm.online_media
  WHERE product_id = 1910;
SELECT product_photo_signature INTO image_sig2 FROM pm.online_media
  WHERE product_id = 1940;
 -- Compare the images:
 value := ORDSYS.ORDImageSignature.isSimilar(image_sig1,
 image_sig2,'color=1.0,texture=0,shape=0,location=0',10);
 IF value = 1 THEN
   DBMS_OUTPUT.PUT_LINE('The images are similar');
 ELSIF value = 0 THEN
   DBMS_OUTPUT.PUT_LINE('The images are not similar');
 END IF;
END;
/
```

ORDImageSignature Operators

The following ORDImageSignature operators are schema level operators and do not reside within a package. These operators use the domain index, if it exists.

- IMGSimilar( ) on page 6-57
- IMGScore( ) on page 6-62

IMGSimilar( )

Format

```
IMGSimilar (sig1 IN ORDSYS.ORDImageSignature,
            sig2 IN ORDSYS.ORDImageSignature,
            weights IN VARCHAR2,
            threshold IN FLOAT
            [ ,referencetoScore IN NUMBER] );
```

Description

Determines whether or not two images match. Specifically, the operator compares the signature of a query image with the signatures of images stored in a table, and determines whether or not the images match, based on the weights and threshold value. This operator returns 1 if the computed distance measure (weighted average) is less than or equal to the threshold value (indicating a match), and returns 0 when the distance between the two images is more than the threshold value.

Parameters

sig1
The signature of the image to which you are comparing the query image. The data type is ORDImageSignature. To use the domain index for the comparison, this first parameter must be the signature column on which the domain index has been created. Otherwise, the database uses the non-indexed implementation of query evaluation.

sig2
The signature of the query or test image. The data type is ORDImageSignature.

weights
A string consisting of matching attribute names followed by values between 0.0 and 1.0. The matching attributes refer to the weights assigned by the user to the different attributes that influence the kind of match. Attributes not specified by the user have a default value of 0.0.

The string can have all or some of the attributes. The value associated with an attribute specifies its relative importance in determining the distance between the signatures. An attribute with a value of 0.0 is not considered at all, while an

attribute with a value of 1.0 is of the highest importance. The weights supplied are normalized prior to processing such that they add up to 1.0, maintaining the ratios that you supplied. At least one of the attributes must have a value greater than 0.0. The attributes are the following:

- color: A value between 0.0 and 1.0 indicating the importance of the feature color.
- texture: A value between 0.0 and 1.0 indicating the importance of the feature texture.
- shape: A value between 0.0 and 1.0 indicating the importance of the feature shape.
- location: A value between 0.0 and 1.0 indicating the importance of the location of the regions in the image. The location weight string cannot be specified alone, it must be used with another weight string.

threshold
The threshold value with which the weighted sum of the distances is to be compared. If the weighted sum is less than or equal to the threshold value, the images are considered to match. The range of this parameter is from 0.0 to 100.0.

referencetoScore
An optional parameter used when ancillary data (score of similarity) is required elsewhere in the query. Set this parameter to the same value here as used in the IMGScore( ) operator. The data type is NUMBER.

Return Value

Returns an integer value of 0 (not similar) or 1 (match).

Pragmas

None.

Exceptions

None.

Usage Notes

Before the IMGSimilar( ) operator can be used, the image signatures must be created with the generateSignature( ) method. Image signatures that have not been initialized will not be evaluated by the IMGSimilar( ) operator. Also, to use the

domain index, the index of type ORDImageIndex must have already been created. See *Oracle interMedia User's Guide* for information on creating and using the index, and for additional performance tips.

The IMGSimilar( ) operator returns Boolean values to indicate if two images match (true, if their image matching score is below the threshold). If you want to know the score value itself, you can use the IMGScore( ) operator in conjunction with the IMGSimilar( ) operator to retrieve the score computed in the IMGSimilar( ) operator.

The IMGSimilar( ) operator is useful when the application needs a simple Yes or No for whether or not two images match. The IMGScore( ) operator is useful when an application wants to make finer distinctions about matching or to perform special processing based on the degree of similarity between images.

Use this operator to take advantage of the increased performance that is possible using image matching with image signature indexes of the underlying tables (as opposed to using the evaluateScore( ) and isSimilar( ) methods).

Examples

Find all images similar to the query image using a threshold value of 25 and the following weights for the visual attributes:

- Color: 0.2
- Texture: 0.1
- Shape: 0.5
- Location: 0.2

This example assumes you already used the generateSignature( ) method to generate a signature for the query image. If an index exists on the signature column, it will be used automatically.

```
DECLARE
 id              NUMBER;
 image           ORDSYS.ORDImage;
 query_signature ORDSYS.ORDImageSignature;

CURSOR getphotos IS
 SELECT product_id, product_photo FROM pm.online_media WHERE
  ORDSYS.IMGSimilar(product_photo_signature, query_signature,
  'color="0.2" texture="0.1" shape="0.5" location="0.2"', 25) = 1;

BEGIN
 SELECT product_photo_signature INTO query_signature
```

```
  FROM pm.online_media
  WHERE product_id =1910;
 OPEN getphotos;
  LOOP
   FETCH getphotos INTO id, image;
   EXIT WHEN getphotos%NOTFOUND;
   DBMS_OUTPUT.PUT_LINE('Image with ID ' || id || ' matches query image.');
  END LOOP;
 CLOSE getphotos;
END;
/
```

Compute the distance between a signature in a temporary LOB and signatures that are stored in the database:

```
DECLARE
  id NUMBER;
  image ORDSYS.ORDIMAGE;
  query_signature ORDSYS.ORDIMAGESIGNATURE;
  query_image ORDSYS.ORDIMAGE;

  CURSOR getphotos IS
   SELECT product_id, product_photo FROM pm.online_media WHERE
      ordsys.IMGsimilar(product_photo_signature, query_signature,
      'color="0.2" texture="0.1" shape="0.5" location="0.2"', 25) = 1;

BEGIN
 SELECT product_photo INTO query_image
  FROM pm.online_media WHERE product_id = 1910;
  -- Initialize signature object
  query_signature := ORDSYS.ORDIMAGESIGNATURE.init();
  -- Create temporary storage for the LOB in the signature object
  DBMS_LOB.CREATETEMPORARY(query_signature.signature, TRUE);
  query_signature.generateSignature(query_image);
  OPEN getphotos;
   LOOP
     FETCH getphotos INTO id, image;
     EXIT WHEN getphotos%NOTFOUND;
     DBMS_OUTPUT.PUT_LINE('Image with ID ' || id || ' matches query image');
   END LOOP;
 CLOSE getphotos;
 DBMS_LOB.FREETEMPORARY(query_signature.signature);
END;
/
```

Generate a signature for an image that is not stored in the database, store the signature in a temporary LOB, and compute the distance between this signature and the signatures that are stored in the database:

```
DECLARE
 id NUMBER;
 image ORDSYS.ORDIMAGE;
 query_signature ORDSYS.ORDIMAGESIGNATURE;
 query_image ORDSYS.ORDIMAGE;
 CURSOR getphotos IS
  SELECT product_id, product_photo FROM pm.online_media WHERE
     ORDSYS.IMGsimilar(product_photo_signature, query_signature,
     'color="0.2" texture="0.1" shape="0.5" location="0.2"', 25) = 1;
BEGIN
  -- Initialize and set the properties for the image object
  query_image := ORDSYS.ORDIMAGE.init('FILE','FILE_DIR','red.gif');
  query_image.setproperties;
  -- Initialize the signature object
  query_signature := ORDSYS.ORDIMAGESIGNATURE.init();
  -- Create temporary storage for the BLOB in the signature object
  DBMS_LOB.CREATETEMPORARY(query_signature.signature, TRUE);
  -- Generate a signature for the query image
  query_signature.generateSignature(query_image);

  OPEN getphotos;
    LOOP
       FETCH getphotos INTO id, image;
       EXIT WHEN getphotos%NOTFOUND;
       DBMS_OUTPUT.PUT_LINE('Image with ID ' || id || ' matches query image');
    END LOOP;
  CLOSE getphotos;
DBMS_LOB.FREETEMPORARY(query_signature.signature);
END;
/
```

IMGScore( )

Format

IMGScore (NUMBER);

Description

Compares the signatures of two images and returns a number representing the weighted sum of the distances for the visual attributes. The IMGScore( ) operator is an ancillary operator, used only in conjunction with the primary operator, IMGSimilar( ), to retrieve the score computed in the IMGSimilar( ) operator. Each IMGScore( ) and IMGSimilar( ) operator shares the same reference number.

Parameters

NUMBER (referencetoSimilar)
Identifier to an IMGSimilar( ) operator. This identifier indicates that the image-matching score value returned by the IMGScore( ) operator is the same one used in the corresponding IMGSimilar( ) operator. This parameter can also be used to maintain references for multiple invocations of the IMGSimilar( ) operator. The data type is NUMBER.

Return Value

This function returns a FLOAT value between 0.0 and 100.0, where 0.0 means the images are identical and 100.0 means the images are completely different.

Pragmas

None.

Exceptions

None.

Usage Notes

Before the IMGScore( ) operator can be used, the image signatures must be created with the generateSignature( ) method. Image signatures that have not been initialized will not be evaluated by the IMGScore( ) operator. Also, if you want the comparison to use the domain index, the index of type ORDImageIndex must have

already been created. See *Oracle interMedia User's Guide* for information on creating and using the index, and for additional performance tips.

The IMGScore( ) operator can be useful when an application wants to make finer distinctions about matching than the simple Yes or No returned by the IMGSimilar( ) operator. For example, using the score returned by the IMGScore( ) operator, the application might assign each image being compared to one of several categories, such as Definite Matches, Probable Matches, Possible Matches, and Nonmatches. The IMGScore( ) operator can also be useful if the application needs to perform special processing based on the degree of similarity between images.

Use this operator to take advantage of the increased performance that is possible using image matching with image signature indexes of the underlying tables (as opposed to using the evaluateScore( ) and isSimilar( ) methods).

Examples

Find the weighted sum of the distances between a test image and the other images in the pm.online_media table, using a threshold of 20 and the following weights for the visual attributes:

- Color: 0.2
- Texture: 0.1
- Shape: 0.5
- Location: 0.2

This example assumes that the signatures were already created using the generateSignature( ) method. Notice that both IMGScore( ) and IMGSimilar( ) are using 123 as the reference number (referred to as the referenceToScore parameter for the IMGSimilar( ) operator) in this example.

```
DECLARE
 id              NUMBER;
 img_score       NUMBER;
 t_img           ORDSYS.ORDImage;
 query_signature ORDSYS.ORDImageSignature;

 CURSOR getphotos IS
 SELECT product_id, ORDSYS.IMGScore(123), product_photo
  FROM pm.online_media WHERE
  ORDSYS.IMGSimilar(product_photo_signature, query_signature,
  'color="0.2" texture="0.1" shape="0.5" location="0.2"', 20, 123) = 1;

BEGIN
```

```
 SELECT product_photo_signature INTO query_signature
   FROM pm.online_media
   WHERE product_id = 1910;
 OPEN getphotos;
  LOOP
   FETCH getphotos INTO id, img_score, t_img;
   EXIT WHEN getphotos%NOTFOUND;
   DBMS_OUTPUT.PUT_LINE('Image with ID '|| id ||
                        ' matches query image with score ' ||
                          img_score ||'.');
  END LOOP;
 CLOSE getphotos;
END;
/
```

The following shows possible results from this example. Image 1910 has a score of 0 because it is the query image; image 1940 is the best match to the query image (besides the query image itself) because it has the lowest score. Changing the weights used in the scoring might lead to different results.

```
Image with ID 1910 matches query image with score 0.
Image with ID 1940 matches query image with score  10.2663.
Image with ID 1743 matches query with with score  15.4666.
```

Demonstrate the use of reference numbers to refer to scores evaluated in different IMGSimilar( ) calls.

In this example, a query is searching for an image in the pm.online_media table that is similar in color to query_sig1, and similar in shape and location to query_sig2.

```
DECLARE
 id              NUMBER;
 img_score1      NUMBER;
 img_score2      NUMBER;
 t_img           ORDSYS.ORDImage;
 query_sig1      ORDSYS.ORDImageSignature;
 query_sig2      ORDSYS.ORDImageSignature;

CURSOR getphotos IS
 SELECT product_id, ORDSYS.IMGScore(1), ORDSYS.IMGScore(2), product_photo
  FROM pm.online_media WHERE
  ORDSYS.IMGSimilar(product_photo_signature,query_sig1,'color="1.0"', 30, 1) = 1
  AND
  ORDSYS.IMGSimilar(product_photo_signature, query_sig2,'shape="0.5"
                   location="0.2"', 30, 2) = 1;

BEGIN
```

```
 SELECT product_photo_signature INTO query_sig1
   FROM pm.online_media
   WHERE product_id = 1910;
 SELECT product_photo_signature INTO query_sig2
   FROM pm.online_media
   WHERE product_id = 1940;
 OPEN getphotos;
  LOOP
   FETCH getphotos INTO id, img_score1, img_score2, t_img;
   EXIT WHEN getphotos%NOTFOUND;
   DBMS_OUTPUT.PUT_LINE('Image with ID '|| id || ' matches the query images.');
  END LOOP;
 CLOSE getphotos;
END;
/
```

7

SQL/MM Still Image

Oracle *inter*Media (*"inter*Media") contains the following information about object types that comply with the first edition of the ISO/IEC 13249-5:2001 SQL MM Part5:StillImage standard (commonly referred to as the SQL/MM Still Image standard):

- SI_AverageColor Object Type on page 7-5

  Describes the average color feature of an image.

- SI_Color Object Type on page 7-15

  Encapsulates color values of a digitized image.

- SI_ColorHistogram Object Type on page 7-20

  Describes the relative frequencies of the colors exhibited by samples of an image.

- SI_FeatureList Object Type on page 7-36

  Describes an image that is represented by a composite feature. The composite feature is based on up to four basic image features (SI_AverageColor, SI_ColorHistogram, SI_PositionalColor, and SI_Texture) and their associated feature weights.

- SI_PositionalColor Object Type on page 7-72

  Describes the positional color feature of an image. Assuming that an image is divided into *n* by *m* rectangles, the positional color feature characterizes an image by the *n* by *m* most significant colors of the rectangles.

- SI_StillImage Object Type on page 7-79

  Represents digital images with inherent image characteristics such as height, width, format, and so on.

- SI_Texture Object Type on page 7-123

  Describes the texture feature of the image characterized by the size of repeating items (coarseness), brightness variations (contrast), and predominant direction (directionality).

A public synonym with the corresponding object type name is created for each of these StillImage object types. Therefore, you do not need to specify the ORDSYS schema name when specifying a StillImage object type.

This chapter also includes the following topics:

- SQL Functions and Procedures on page 7-3

  Provides an overview of how the SQL functions and procedures are presented in this guide, as well as how they are created.

- Example Media Table and User Definition on page 7-4

  Presents the media table definition upon which the examples in this chapter are based. The examples in this chapter assume that the test media table SI_MEDIA has been created and filled with data.

- Views on page 7-130

  Describes the views in the SI_INFORMTN_SCHEMA that you can query for information about the supported image formats and implementation-defined values.

- Internal Helper Types on page 7-133

  Provides syntax for attributes that are VARRAY types.

See *Oracle interMedia User's Guide* for a list of ORDImage features that are not available for StillImage objects because the SQL/MM Still Image standard does not specify them.

SQL Functions and Procedures

For each Still Image constructor or method, there is an equivalent SQL function or procedure. Each function or procedure is presented with its equivalent constructor or method. Although the description, parameters, usage notes, and exceptions subsections frequently refer to the method, these subsections are also applicable to the equivalent SQL function or procedure.

All SQL functions and procedures are created as standalone functions in the ORDSYS schema with invoker rights. A public synonym with the corresponding function or procedure name is created for all SQL functions and procedures. Therefore, you do not need to specify the schema name when a function or procedure is called. For example:

Use ORDSYS.SI_MkAvgClr(averageColor) to make the call without the synonym.

Use SI_MkAvgClr(averageColor) to make the call with the synonym.

All database users can call these functions and procedures.

Example Media Table and User Definition

The methods described in this reference chapter show examples based on a media table SI_MEDIA. Refer to the SI_MEDIA table definition that follows when reading through the examples. Before using methods, you will need to load some data into the table using one of the examples provided with the reference information for each object type's constructors.

SI_MEDIA Table Definition

```
CREATE TABLE PM.SI_MEDIA(
  PRODUCT_ID NUMBER(6),
  PRODUCT_PHOTO SI_StillImage,
  AVERAGE_COLOR SI_AverageColor,
  COLOR_HISTOGRAM SI_ColorHistogram,
  FEATURE_LIST SI_FeatureList,
  POSITIONAL_COLOR SI_PositionalColor,
  TEXTURE SI_Texture,
  CONSTRAINT id_pk PRIMARY KEY (PRODUCT_ID));
COMMIT;
```

PM.IMAGETAB Table Definition

Create this table in preparation for the Example 2 for the SI_StillImage(content) method, which begins on page 7-83:

```
CREATE TABLE PM.IMAGETAB (id number, imgblob blob);
COMMIT;
```

User Ron Definition

For a user "ron" to use the examples, the following statements must be issued before ron executes the examples, where "/mydir" is the directory where ron will find the audio, video, and image data:

```
CREATE USER ron IDENTIFIED BY ron;
GRANT INSERT, UPDATE, SELECT, DELETE on OE.ORDERS to ron;
GRANT INSERT, UPDATE, SELECT, DELETE on PM.ONLINE_MEDIA to ron;
GRANT EXECUTE on SYS.DBMS_LOB to ron;
GRANT CREATE SESSION TO ron;
CREATE OR REPLACE DIRECTORY FILE_DIR as '/mydir';
GRANT READ ON DIRECTORY FILE_DIR TO PUBLIC WITH GRANT OPTION;
```

SI_AverageColor Object Type

The SI_AverageColor object type describes the average color feature of an image. It is created in the ORDSYS schema with invoker rights. It is declared as an INSTANTIABLE and NOT FINAL type.

> **Note:** Use the SI_AverageColor object type constructors and method rather than accessing attributes directly to protect yourself from changes to the internal representation of the SI_AverageColor object.

The AverageColor object type is defined as follows:

```
CREATE OR REPLACE TYPE SI_AverageColor
  AUTHID CURRENT_USER
  AS OBJECT
 (
-------------------
-- TYPE ATTRIBUTES
-------------------
SI_AverageColorSpec SI_Color,
---------------------
-- METHOD DECLARATION
---------------------
-- CONSTRUCTORS
--
CONSTRUCTOR FUNCTION SI_AverageColor
(sourceImage IN SI_StillImage)
RETURN SELF AS RESULT DETERMINISTIC,

CONSTRUCTOR FUNCTION SI_AverageColor
(SI_AverageColorSpec IN SI_Color)
     return SELF AS RESULT DETERMINISTIC,

-- Methods associated with the source attribute
MEMBER FUNCTION SI_Score
(image in SI_StillImage)
 RETURN DOUBLE PRECISION DETERMINISTIC
) INSTANTIABLE
  NOT FINAL;
/
```

where:

- SI_AverageColorSpec: the average color of the object.

SI_AverageColor Constructors

This section describes the SI_AverageColor object constructors, which are the following:

SI_AverageColor(averageColorSpec)

Format

```
SI_AverageColor(averageColorSpec IN SI_Color)
 RETURN SELF AS RESULT DETERMINISTIC;
```

Format of Equivalent SQL Function

```
SI_MkAvgClr(avgClr IN SI_Color) RETURN SI_AverageColor DETERMINISTIC;
```

Description

Constructs an SI_AverageColor object. The SI_AverageColorSpec attribute is initialized with the value of the specified color.

Parameters

averageColorSpec
avgClr
The color used to construct an SI_AverageColor object.

Pragmas

None.

Exceptions

None.

Usage Notes

An error message is returned if one or more of the following conditions are true:

- The value of the specified averageColorSpec is NULL.
- The value of the specified averageColorSpec is not a valid SI_Color value.

Examples

Construct an SI_AverageColor object from a specified color using the SI_AverageColor(averageColorSpec) constructor:

```
DECLARE
   myColor SI_Color;
```

```
   myAvgColor SI_AverageColor;
BEGIN
 myColor := NEW SI_COLOR(null, null, null);
 myColor.SI_RGBColor(10, 100, 200);
 myAvgColor := NEW SI_AverageColor(myColor);
 INSERT INTO PM.SI_MEDIA (product_id, average_color) VALUES (75, myAvgColor);
 COMMIT;
END;
/
```

Construct an SI_AverageColor object from a specified color using the SI_MkAvgClr( ) function:

```
DECLARE
   myColor SI_Color;
   myAvgColor SI_AverageColor;
BEGIN
   myColor := NEW SI_COLOR(null, null, null);
   myColor.SI_RGBColor(10, 100, 200);
   myAvgColor := SI_MkAvgClr(myColor);
   INSERT INTO PM.SI_MEDIA (product_id, average_color)
     VALUES (89, myAvgColor);
   COMMIT;
END;
/
```

SI_AverageColor(sourceImage)

Format

```
SI_AverageColor(sourceImage IN SI_StillImage)
 RETURN SELF AS RESULT DETERMINISTIC;
```

Format of Equivalent SQL Function

```
SI_FindAvgClr(sourceImage IN SI_StillImage)  RETURN SI_AverageColor DETERMINISTIC;
```

Description

Derives an SI_AverageColor value from the specified image. The image is divided into *n* samples. Then, each component (red, green, blue) of all the samples is added separately and divided by the number of samples. This gives the values of the components of the specified image. The process by which SI_AverageColor is determined can also be described by the following expression, where *n* is the number of samples:

$$\left(\frac{\sum_{i=1}^{n} \text{red value}}{n} , \frac{\sum_{i=1}^{n} \text{green value}}{n} , \frac{\sum_{i=1}^{n} \text{blue value}}{n} \right)$$

Parameters

sourceImage
The image from which the average color feature is extracted.

Pragmas

None.

Exceptions

None.

Usage Notes

An error is returned if one or more of the following conditions are true:

- The value of the specified image is NULL.

- The value of sourceImage.SI_Content is NULL.
- The average color feature is not supported for the format of the specified image. This is determined by looking up the SI_IMAGE_FORMAT_FEATURES view or SI_IMAGE_FRMT_FTRS view.

Examples

Derive an SI_AverageColor value using the SI_AverageColor(sourceImage) constructor:

```
DECLARE
   myimage SI_StillImage;
   myAvgColor SI_AverageColor;
BEGIN
   SELECT product_photo INTO myimage FROM PM.SI_MEDIA WHERE product_id=1;
   myAvgColor := NEW SI_AverageColor(myimage);
END;
/
```

Derive an SI_AverageColor object from an image using the SI_FindAvgClr( ) function:

```
DECLARE
   myimage SI_StillImage;
   myAvgColor SI_AverageColor;
BEGIN
   SELECT product_photo INTO myimage FROM PM.SI_MEDIA WHERE product_id=1;
   myAvgColor := SI_FindAvgClr(myimage);
END;
/
```

SI_AverageColor Method

This section presents reference information on the SI_AverageColor method used for image matching:

SI_Score( ) for SI_AverageColor

Formats

SI_Score(image in SI_StillImage)

RETURN DOUBLE PRECISION DETERMINISTIC;

Format of Equivalent SQL Function

SI_ScoreByAvgClr(feature IN SI_AverageColor, image IN SI_StillImage)

RETURN DOUBLE PRECISION DETERMINISTIC;

Description

Determines and returns the score of the specified image as compared to the SI_AverageColor object instance to which you apply the method. This method returns a DOUBLE PRECISION value between 0 and 100. A value of 0 indicates that the average color of the specified image and the SI_AverageColor object instance are identical. A value of 100 indicates that average color of the specified image and the SI_AverageColor object instance are completely different.

Parameters

image
The image whose average color feature is compared with the SI_AverageColor object instance to which you apply this method.

feature
An SI_AverageColor value.

Usage Notes

This method returns a NULL value if any of the following is true:

- The value of the SI_AverageColor to which the method is applied is NULL.
- The value of the specified image is NULL.
- The value of image.content_SI is NULL.
- The SI_AverageColor feature is not supported for the specified image format.

Pragmas

None.

Exceptions

None.

Examples

Compare an image to an SI_AverageColor object and return the score using the SI_Score( ) method:

```
 DECLARE
   score DOUBLE PRECISION;
   myimage SI_StillImage;
   myotherimage SI_StillImage;
   myAvgColor SI_AverageColor;
BEGIN
   SELECT product_photo INTO myimage FROM PM.SI_MEDIA WHERE product_id=1;
   myAvgColor := NEW SI_AverageColor(myimage);
   SELECT product_photo INTO myotherimage FROM PM.SI_MEDIA
      WHERE product_id=2;
   score := myAvgColor.SI_Score(myotherimage);
   DBMS_OUTPUT.PUT_LINE('Score is ' || score);
END;
/
```

Compare an image to an SI_AverageColor object and return the score using the SI_ScoreByAvgClr( ) function:

```
DECLARE
    score DOUBLE PRECISION;
    myimage SI_StillImage;
    myotherimage SI_StillImage;
    myAvgColor SI_AverageColor;
BEGIN
   SELECT product_photo INTO myimage FROM PM.SI_MEDIA WHERE product_id=1;
   myAvgColor := NEW SI_AverageColor(myimage);
   SELECT product_photo INTO myotherimage FROM PM.SI_MEDIA WHERE product_id=2;
   score := SI_ScoreByAvgClr(myAvgColor, myotherimage);
   DBMS_OUTPUT.PUT_LINE('Score is ' || score);
END;
/
```

SI_Color Object Type

The SI_Color object type represents color values of a digitized image as an RGB color value. It is created in the ORDSYS schema with invoker rights. It is declared as an INSTANTIABLE and NOT FINAL type.

> **Note:** Use the SI_Color method rather than accessing attributes directly to protect yourself from changes to the internal representation of the SI_Color object.

The SI_Color object is defined as follows:

```
CREATE OR REPLACE TYPE SI_Color
  AUTHID CURRENT_USER
  AS OBJECT
 (
-------------------
-- TYPE ATTRIBUTES
-------------------
redValue   INTEGER,
greenValue INTEGER,
blueValue  INTEGER,
---------------------
-- METHOD DECLARATION
---------------------
MEMBER PROCEDURE SI_RGBColor
 (redValue   IN INTEGER,
  greenValue IN INTEGER,
 blueValue  IN INTEGER)
) INSTANTIABLE
NOT FINAL;
```

where:

- redValue: the integer value of the red component of the RGB color value.
- greenValue: the integer value of the green component of the RGB color value.
- blueValue: the integer value of the blue component of the RGB color value.

SI_Color Constructor

Only a system-default constructor is provided for the SI_Color object.

SI_Color Method

This section presents reference information on the SI_Color method used for constructing an SI_Color object using RGB color values:

SI_RGBColor( )

Format

```
SI_RGBColor(redValue     IN INTEGER,
            greenValue   IN INTEGER,
            blueValue    IN INTEGER);
```

Format of Equivalent SQL Function

```
SI_MkRGBClr(redValue     IN INTEGER,
            greenValue IN INTEGER,
            blueValue IN INTEGER)
 RETURN SI_Color;
```

Description

Constructs an SI_Color object in the RGB color space using the specified red, blue, and green values.

Parameters

redValue
An integer value between 0 and 255.

greenValue
An integer value between 0 and 255.

blueValue
An integer value between 0 and 255.

Usage Notes

- You must call the system default constructor and specify NULL values, then call the SI_RGBColor method to set the RGB values. This two-step process is required because:
  - The SQL/MM standard does not specify an object constructor for this type, therefore, the need to use the system default constructor.

  - The default constructor does not perform any argument validation. By calling the SI_RGBColor method, specified values will be validated before assigning them to the color attributes.
- An error is returned if any of the specified values is NULL or if any of the specified values is not between 0 and 255.

Pragmas

None.

Exceptions

None.

Examples

Construct an SI_Color value using the SI_RGBColor( ) method:

```
DECLARE
   myColor SI_Color;
BEGIN
   myColor := NEW SI_COLOR(null, null, null);
   -- Set myColor to represent the color red:
   myColor.SI_RGBColor(255, 0, 0);
END;
/
```

Construct an SI_Color value using the SI_MkRGBClr ( ) function:

```
DECLARE
   myColor SI_Color;
BEGIN
   myColor := NEW SI_COLOR(null, null, null);
   -- Set myColor to represent the color red:
   myColor := SI_MkRGBClr(255, 0, 0);
END;
/
```

SI_ColorHistogram Object Type

The SI_ColorHistogram object represents the color histogram image feature. It describes the relative frequencies of the colors exhibited by samples of an image. It is created in the ORDSYS schema with invoker rights. It is declared as an INSTANTIABLE and NOT FINAL type. This object type is defined as follows. (See "Internal Helper Types" on page 7-133 for the colorsList and colorFrequenciesList attribute syntax.)

> **Note:** Use the SI_ColorHistogram constructors and methods rather than accessing attributes directly to protect yourself from changes to the internal representation of the SI_ColorHistogram object.

The SI_ColorHistogram object is defined as follows:

```
CREATE OR REPLACE TYPE SI_ColorHistogram
  AUTHID CURRENT_USER
  AS OBJECT
  (
-------------------
-- TYPE ATTRIBUTES
-------------------
SI_ColorsList       colorsList,
SI_FrequenciesList  colorFrequenciesList,
--------------------
-- METHOD DECLARATION
--------------------
-- CONSTRUCTORS
CONSTRUCTOR FUNCTION SI_ColorHistogram
    (sourceImage IN SI_StillImage)
     RETURN SELF AS RESULT DETERMINISTIC,
CONSTRUCTOR FUNCTION SI_ColorHistogram
    (firstColor IN SI_Color,
     frequency  IN DOUBLE PRECISION)
     RETURN SELF AS RESULT DETERMINISTIC,
CONSTRUCTOR FUNCTION SI_ColorHistogram
    (SI_ColorsList       IN colorsList,
     SI_FrequenciesList  IN colorFrequenciesList)
     RETURN SELF AS RESULT DETERMINISTIC,
MEMBER PROCEDURE SI_Append
```

```
    (color     IN SI_Color,
     frequency IN DOUBLE PRECISION),
MEMBER FUNCTION SI_Score
     (image    IN SI_StillImage)
RETURN DOUBLE PRECISION DETERMINISTIC
) INSTANTIABLE
  NOT FINAL;
/
```

where:

- SI_ColorsList: array of the SI_Color object type that represents the color values of the image.
- SI_FrequenciesList: array of the colorFrequencies attribute that represents the color frequencies of the image. Its values range from 0 to 100. Each array element represents the frequency of the corresponding color in the SI_ ColorsList array.

SI_ColorHistogram Constructors

This section describes the SI_ColorHistogram object constructors, which are the following:

SI_ColorHistogram(colors, frequencies)

Format

```
SI_ColorHistogram(SI_ColorsList          IN colorsList,
                  SI_FrequenciesList  IN colorFrequenciesList)
    RETURN SELF AS RESULT DETERMINISTIC;
```

Format of Equivalent SQL Function

```
SI_ArrayClrHstgr(colors IN SI_ColorsList,
                 frequencies IN colorFrequenciesList),
RETURN SI_ColorHistogram DETERMINISTIC;
```

Description

Constructs an SI_ColorHistogram object. The following attributes are initialized:

- The SI_ColorsList array attribute is initialized with the value of the specified colors.
- The SI_FrequenciesList array attribute is initialized with the value of the specified frequencies.

See "Internal Helper Types" on page 7-133 for the SI_ColorsList and colorFrequenciesList attribute syntax.

Parameters

SI_ColorsList
colors
An array of colors with a maximum size of SI_MaxHistogramLength. Query the SI_VALUES view in SI_INFORMTN_SCHEMA for the value of SI_MaxHistogramLength.

SI_FrequenciesList
frequencies
An array of color frequencies with a maximum size of SI_MaxHistogramLength.

Pragmas

None.

Exceptions

None.

Usage Notes

An error is returned if any one of the following conditions is true:

- Any one of the specified values is NULL.
- Any one of the specified frequency values is less than 0 or greater than 100.

Examples

Construct an SI_ColorHistogram object using the SI_ColorHistogram(colors, frequencies) constructor:

```
 DECLARE
  myColors ordsys.colorsList;
  myFreqs  ordsys.colorFrequenciesList;
  myColorHist SI_ColorHistogram;
BEGIN
    --Create the varray objects colors and frequencies:
    myColors := ordsys.colorsList();
    myFreqs :=  ordsys.colorFrequenciesList();
    --Create 100 empty elements in the varray:
    myColors.extend(100);
    myFreqs.extend(100);
    --Add three colors to the varray:
    myColors :=  ordsys.colorslist(SI_mkRGBClr(10, 20, 255),
                           SI_mkRGBClr(200,200,255),
                           SI_mkRGBClr(35,100,100));
    --Add three frequencies to the varray:
    myFreqs := ordsys.colorFrequenciesList(10, 1, 90);
    --Construct the histogram using the colors and frequency varrays:
    myColorHist := NEW SI_ColorHistogram(myColors, myFreqs);
END;
/
```

Construct an SI_ColorHistogram object using the SI_ArrayClrHstgr( ) function:

```
DECLARE
  myColors ordsys.colorsList;
  myFreqs  ordsys.colorFrequenciesList;
  myColorHist SI_ColorHistogram;
BEGIN
    --Create the varray objects colors and frequencies:
```

```
    myColors := ordsys.colorsList();
    myFreqs :=  ordsys.colorFrequenciesList();
    --Create 100 empty elements in the varray:
    myColors.extend(100);
    myFreqs.extend(100);
    --Add three colors to the varray:
    myColors :=  ordsys.colorslist(SI_mkRGBClr(10, 20, 255),
                          SI_mkRGBClr(200,200,255),
                          SI_mkRGBClr(35,100,100));
    --Add three frequencies to the varray:
    myFreqs := ordsys.colorFrequenciesList(10, 1, 90);
    --Construct the histogram using the colors and frequency varrays:
    myColorHist := SI_ArrayClrHstgr(myColors, myFreqs);
END;
/
```

SI_ColorHistogram(firstColor, frequency)

Format

```
SI_ColorHistogram(firstColor IN SI_Color,
                  frequency  IN DOUBLE PRECISION)
   RETURN SELF AS RESULT DETERMINISTIC;
```

Format of the Equivalent SQL Function

```
SI_MkClrHstgr(firstColor IN SI_Color, frequency IN DOUBLE PRECISION)
   RETURN SI_ColorHistogram DETERMINISTIC;
```

Description

Creates a single color/frequency pair in an SI_ColorHistogram object. The following attributes are initialized:

- The SI_ColorsList array attribute is initialized with the value of the specified firstColor.
- The SI_FrequenciesList array attribute is initialized with the value of the specified frequency.

Parameters

firstColor
A color value of SI_ColorHistogram.

frequency
The frequency value of SI_ColorHistogram for the firstColor parameter.

Pragmas

None.

Exceptions

None.

Usage Notes

An error is returned if any of the following conditions is true:

- Any one of the specified values is NULL.
- The frequency specified is less than 0 or greater than 100.

Examples

Create a single color/frequency pair for an SI_ColorHistogram object using the SI_ColorHistogram(firstColor, frequency) constructor:

```
DECLARE
   myColor  SI_Color;
   myClrHist SI_ColorHistogram;
BEGIN
   -- Initialize myColor:
   myColor := new SI_Color(null, null, null);
   myColor.SI_RGBColor(10, 45, 200);
   -- Construct the histogram using myColor and a frequency of 10.0:
   myClrHist := new SI_ColorHistogram(myColor, 10.0);
END;
/
```

Construct a single color/frequency pair for an SI_ColorHistogram object using the SI_MkClrHstgr( ) function:

```
DECLARE
   myColor  SI_Color;
   myClrHist SI_ColorHistogram;
BEGIN
   -- Initialize myColor:
   myColor := new SI_Color(null, null, null);
   myColor.SI_RGBColor(10, 45, 200);
   -- Construct the histogram using myColor and a frequency of 10.0:
   myClrHist := SI_MkClrHstgr(myColor, 10.0);
END;
/
```

SI_ColorHistogram(sourceImage)

Format

```
SI_ColorHistogram(sourceImage IN SI_StillImage)
RETURN SELF AS RESULT DETERMINISTIC;
```

Format of Equivalent SQL Function

```
SI_FindClrHstgr (sourceImage IN SI_StillImage)
   RETURN SI_ColorHistogram DETERMINISTIC;
```

Description

Extracts a color histogram from the specified image. The following attributes are initialized:

- The SI_ColorsList attribute is initialized with the color values derived from the specified image.
- The SI_FrequenciesList attribute is initialized with the frequencies derived from the specified image.

Parameters

sourceImage
The image from which the color histogram is extracted.

Pragmas

None.

Exceptions

None.

Usage Notes

An error is returned if any of the following conditions is true:

- The value of the specified image is NULL.
- The value of sourceImage.SI_Content is NULL.
- The color histogram feature is not supported for this image format.

To determine whether or not the color histogram feature is supported for a given image format, query the SI_IMAGE_FORMAT_FEATURES view or SI_IMAGE_FRMT_FTRS view.

Examples

Extract an SI_ColorHistogram object from an image using the SI_ColorHistogram(sourceImage) constructor:

```
DECLARE
   myColorHist SI_ColorHistogram;
   myimage    SI_StillImage;
BEGIN
 -- Select a product_photo from the si_media table:
 SELECT product_photo INTO myimage FROM PM.SI_MEDIA WHERE product_id=2 FOR
 UPDATE;
 -- Extract the color histogram from the source image:
  myColorHist := NEW SI_ColorHistogram(myimage);
 -- Update the color_histogram column with the extracted value:
 UPDATE PM.SI_MEDIA SET color_histogram = myColorHist WHERE product_id=2;
 COMMIT;
END;
/
```

Extract an SI_ColorHistogram object from an image using the SI_FindClrHstgr( ) function:

```
DECLARE
   myColorHist SI_ColorHistogram;
   myimage    SI_StillImage;
BEGIN
 SELECT product_photo INTO myimage FROM PM.SI_MEDIA WHERE product_id=2 FOR
 UPDATE;
   myColorHist := SI_FindClrHstgr(myimage);
   UPDATE PM.SI_MEDIA SET color_histogram = myColorHist WHERE product_id=2;
 COMMIT;
END;
/
```

SI_ColorHistogram Methods

This section presents reference information on the SI_ColorHistogram methods used for color histogram construction and image matching, which are the following:

SI_Append( )

Format

```
SI_Append(color       IN SI_Color,
          frequency IN DOUBLE PRECISION);
```

Format of Equivalent SQL Procedure

```
SI_AppendClrHstgr(feature     IN OUT NOCOPY SI_ColorHistogram,
                  color       IN SI_Color,
                  frequency IN DOUBLE PRECISION);
```

Description

Extends a specified SI_ColorHistogram value by a color/frequency pair.

Parameters

color
The color value to be added to the histogram.

frequency
The corresponding frequency value of the specified color that is to be added to the histogram.

feature
The color histogram to which the color and frequency values are appended.

Usage Notes

An error is returned if any one of the following conditions is true:

- Any of the specified values is NULL.
- The frequency is less than 0 or greater than 100.
- The attribute SI_ColorsList already has SI_MaxHistogramLength elements.

Pragmas

None.

Exceptions

None.

Examples

Extend an SI_ColorHistogram value by a color/frequency pair using the SI_Append( ) method:

```
DECLARE
   myColor  SI_Color;
   myClrHist SI_ColorHistogram;
BEGIN
   -- Initialize myColor:
   myColor := new SI_Color(null, null, null);
   myColor.SI_RGBColor(10, 45, 200);
   -- Construct a histogram using a single color/frequency pair
   -- consisting of myColor and a frequency of 10.0:
   myClrHist := new SI_ColorHistogram(myColor, 10.0);
   -- Append myClrHist with another color/frequency pair:
   myColor.SI_RGBColor(60, 55, 100);
   myClrHist.SI_Append(myColor, 20.0);
END;
/
```

Extend an SI_ColorHistogram value by a color/frequency pair using the SI_AppendClrHstgr( ) procedure:

```
DECLARE
   myColor  SI_Color;
   myClrHist SI_ColorHistogram;
BEGIN
   -- Initialize myColor:
   myColor := new SI_Color(null, null, null);
   myColor.SI_RGBColor(10, 45, 200);
   -- Construct a histogram using a single color/frequency pair
   -- consisting of myColor and a frequency of 10.0:
   myClrHist := new SI_ColorHistogram(myColor, 10.0);
   -- Append myClrHist with another color/frequency pair:
   myColor.SI_RGBColor(60, 55, 100);
   SI_AppendClrHstgr(myClrHist, myColor, 20.0);
END;
/
```

SI_Score( ) for SI_ColorHistogram

Format

SI_Score(image IN SI_StillImage)

RETURN DOUBLE PRECISION DETERMINISTIC;

Format of Equivalent SQL Function

SI_ScoreByClrHstgr(feature IN SI_ColorHistogram,

image IN SI_StillImage) RETURN DOUBLE PRECISION DETERMINISTIC;

Description

Determines and returns the score of the color histogram of the specified image as compared to the SI_ColorHistogram object instance to which you apply this method. This method returns a DOUBLE PRECISION value between 0 and 100. A value of 0 means that the color histogram of the specified image and the SI_ ColorHistogram object instance are identical. A value of 100 indicates that the color histogram of the specified image and the SI_ColorHistogram object instance are completely different. A NULL value is returned if any one of the following is true:

- The value of the SI_ColorHistogram object instance is NULL.
- The value of the specified image is NULL.
- The value of image.SI_Content is NULL.
- The value of the color histogram feature is not supported for the format of the specified image.

Parameters

image
The image whose color histogram feature is extracted and used for comparison.

feature
The histogram to be compared with the color histogram of the specified image.

Usage Notes

None.

Pragmas

None.

Exceptions

None.

Examples

Compare a color histogram of an image to an SI_ColorHistogram value and return the score using the SI_Score method:

```
DECLARE
   myClrHistConstructed SI_ColorHistogram;
   myImage              SI_StillImage;
   myColor              SI_Color;
   score DOUBLE PRECISION;
BEGIN
   -- Get a stored image:
   SELECT product_photo INTO myimage FROM PM.SI_MEDIA WHERE product_id=2;
   -- Initialize myColor
   myColor := new SI_Color(null, null, null);
   myColor.SI_RGBColor(10, 45, 200);
   -- Construct a histogram using a single color/frequency pair
   -- consisting of myColor and a frequency of 10.0:
   myClrHistConstructed := new SI_ColorHistogram(myColor, 10.0);
   -- Compare the color histogram of the stored image to myClrHistConstructed:
   score := myClrHistConstructed.SI_Score(myImage);
   DBMS_OUTPUT.PUT_LINE('Score is ' || score);
END;
/
```

Compare the color histogram of an image to an SI_ColorHistogram value and return the score using the SI_ScoreByClrHstgr( ) function:

```
DECLARE
   myClrHistConstructed SI_ColorHistogram;
   myImage              SI_StillImage;
   myColor              SI_Color;
   score DOUBLE PRECISION;
BEGIN
   -- Get a stored image
   SELECT product_photo INTO myimage FROM PM.SI_MEDIA WHERE product_id=2;
   -- Initialize myColor:
   myColor := new SI_Color(null, null, null);
```

```
   myColor.SI_RGBColor(10, 45, 200);
   -- Construct a histogram using a single color/frequency pair
   -- consisting of myColor and a frequency of 10.0:
   myClrHistConstructed := new SI_ColorHistogram(myColor, 10.0);
   -- Compare the color histogram of the stored image to myClrHistConstructed:
   score  := SI_ScoreByClrHstgr(myClrHistConstructed, myImage);
   DBMS_OUTPUT.PUT_LINE('Score is ' || score);
END;
/
```

SI_FeatureList Object Type

A composite feature that contains up to four different basic features and their associated feature weights. A weight value specifies the importance given to a particular feature during image matching. Each weight value can have a value from 0.0 and 1.0. A feature weight value of 0.0 indicates that the feature is not considered for image matching. This object type is created in the ORDSYS schema with invoker rights. It is declared as an INSTANTIABLE and NOT FINAL type.

> **Note:** Use the SI_FeatureList constructor and methods rather than accessing attributes directly to protect yourself from changes to the internal representation of the SI_FeatureList object.

The SI_FeatureList object type is defined as follows:

```
CREATE OR REPLACE TYPE SI_FeatureList
  AUTHID CURRENT_USER
  AS OBJECT
  (
   --attributes
   AvgClrFtr_SI         SI_AverageColor,
   AvgClrFtrWght_SI     DOUBLE PRECISION,
   ClrHstgrFtr_SI       SI_ColorHistogram,
   ClrHstgrFtrWght_SI   DOUBLE PRECISION,
   PstnlClrFtr_SI       SI_PositionalColor,
   PstnlClrFtrWght_SI   DOUBLE PRECISION,
   TextureFtr_SI        SI_Texture,
   TextureFtrWght_SI    DOUBLE PRECISION,
     --
     --
     --Methods
CONSTRUCTOR FUNCTION SI_FeatureList
     (AvgClrFtr_SI             IN  SI_AverageColor,
      AvgClrFtrWght_SI         IN  DOUBLE PRECISION,
      ClrHstgrFtr_SI           IN  SI_ColorHistogram,
      ClrHstgrFtrWght_SI       IN  DOUBLE PRECISION,
      PstnlClrFtr_SI           IN  SI_PositionalColor,
      PstnlClrFtrWght_SI       IN  DOUBLE PRECISION,
      TextureFtr_SI            IN  SI_Texture,
      TextureFtrWght_SI        IN  DOUBLE PRECISION)
     return SELF AS RESULT DETERMINISTIC,
```

```
     --
     --
MEMBER PROCEDURE SI_SetFeature
     (averageColorFeature         IN  SI_AverageColor,
     averageColorFeatureWeight    IN  DOUBLE PRECISION),
     --
MEMBER PROCEDURE SI_SetFeature
    (colorHistogramFeature        IN  SI_ColorHistogram,
     colorHistogramFeatureWeight  IN  DOUBLE PRECISION),
    --
MEMBER PROCEDURE SI_SetFeature
    (positionalColorFeature       IN  SI_PositionalColor,
     positionalColorFeatureWeight IN  DOUBLE PRECISION),
    --
MEMBER PROCEDURE SI_SetFeature
    (textureFeature               IN  SI_Texture,
     textureFeatureWeight         IN  DOUBLE PRECISION),
     --
MEMBER FUNCTION SI_Score
     (image IN SI_StillImage)
      RETURN DOUBLE PRECISION DETERMINISTIC,
      --
MEMBER FUNCTION SI_AvgClrFtr( )
     RETURN SI_AverageColor DETERMINISTIC,
     PRAGMA RESTRICT_REFERENCES(SI_AvgClrFtr, WNDS, WNPS, RNDS, RNPS),
     --
MEMBER FUNCTION SI_AvgClrFtrWght( )
     RETURN DOUBLE PRECISION DETERMINISTIC,
     PRAGMA RESTRICT_REFERENCES(SI_AvgClrFtrWght, WNDS, WNPS, RNDS, RNPS),
     --
MEMBER FUNCTION SI_ClrHstgrFtr( )
     RETURN SI_ColorHistogram DETERMINISTIC,
     PRAGMA RESTRICT_REFERENCES(SI_ClrHstgrFtr, WNDS, WNPS, RNDS, RNPS),
     --
MEMBER FUNCTION SI_ClrHstgrFtrWght( )
     RETURN DOUBLE PRECISION DETERMINISTIC,
     PRAGMA RESTRICT_REFERENCES(SI_ClrHstgrFtrWght, WNDS, WNPS, RNDS, RNPS),
     --
MEMBER FUNCTION SI_PstnlClrFtr( )
     RETURN SI_PositionalColor DETERMINISTIC,
     PRAGMA RESTRICT_REFERENCES(SI_PstnlClrFtr, WNDS, WNPS, RNDS, RNPS),
     --
MEMBER FUNCTION SI_PstnlClrFtrWght( )
     RETURN DOUBLE PRECISION DETERMINISTIC,
     PRAGMA RESTRICT_REFERENCES(SI_PstnlClrFtrWght, WNDS, WNPS, RNDS, RNPS),
```

```
     --
MEMBER FUNCTION SI_TextureFtr( )
     RETURN SI_Texture DETERMINISTIC,
     PRAGMA RESTRICT_REFERENCES(SI_TextureFtr, WNDS, WNPS, RNDS, RNPS),
     --
MEMBER FUNCTION SI_TextureFtrWght( )
   RETURN DOUBLE PRECISION DETERMINISTIC,
   PRAGMA RESTRICT_REFERENCES(SI_TextureFtrWght, WNDS, WNPS, RNDS, RNPS)
   --
)
INSTANTIABLE
NOT FINAL;
/
```

where:

- AvgClrFtr_SI: average color.
- AvgClrFtrWght_SI: average color feature weight with a default value of 0.0.
- ClrHstgrFtr_SI: color histogram.
- ClrHstgrFtrWght_SI: color histogram weight with a default value of 0.0.
- PstnlClrFtr_SI: positional color.
- PstnlClrFtrWght_SI: positional color weight with a default value of 0.0.
- TextureFtr_SI: texture.
- TextureFtrWght_SI: texture weight with a default value of 0.0.

SI_FeatureList Constructor

This section describes the SI_FeatureList constructor.

The SI_FeatureList constructor is as follows:

- SI_FeatureList( ) on page 7-40

SI_FeatureList( )

Format

```
SI_FeatureList((AvgClrFtr_SI IN  SI_AverageColor,
AvgClrFtrWght_SI             IN  DOUBLE PRECISION,
ClrHstgrFtr_SI               IN  SI_ColorHistogram,
ClrHstgrFtrWght_SI           IN  DOUBLE PRECISION,
PstnlClrFtr_SI               IN  SI_PositionalColor,
PstnlClrFtrWght_SI           IN  DOUBLE PRECISION,
TextureFtr_SI                IN  SI_Texture,
TextureFtrWght_SI            IN  DOUBLE PRECISION)
```

Format of Equivalent SQL Function

```
SI_MkFtrList(averageColorFeature IN SI_AverageColor,
averageColorFeatureWeight        IN DOUBLE PRECISION,
colorHistogramFeature            IN SI_ColorHistogram,
colorHistogramFeatureWeight      IN DOUBLE PRECISION,
positionalColorFeature           IN SI_PositionalColor,
positionalColorFeatureWeight     IN DOUBLE PRECISION,
textureFeature                   IN SI_Texture,
textureFeatureWeight             IN  DOUBLE PRECISION)
RETURN SI_FeatureList;
```

Description

Constructs an SI_FeatureList object. All the feature and feature weight attributes are set to the corresponding values of the input parameters.

Parameters

AvgClrFtr_SI
averageColorFeature
The average color of SI_FeatureList.

AvgClrFtrWght_SI
averageColorFeatureWeight
The average color weight of SI_FeatureList. The default value is 0.0. The weight value can range from 0.0 to 1.0. A value of 0.0 indicates that the feature should not be considered during image matching.

ClrHstgrFtr_SI
colorHistogramFeature
The color histogram of SI_FeatureList.

ClrHstgrFtrWght_SI
colorHistogramFeatureWeight
The color histogram weight of SI_FeatureList. The default value is 0.0. The weight value can range from 0.0 to 1.0. A value of 0.0 indicates that the feature should not be considered during image matching.

PstnlClrFtr_SI
positionalColorFeature
The positional color of SI_FeatureList.

PstnlClrFtrWght_SI
positionalColorFeatureWeight
The positional color weight of SI_FeatureList. The default value is 0.0. The weight value can range from 0.0 to 1.0. A value of 0.0 indicates that the feature should not be considered during image matching.

TextureFtr_SI
textureFeature
The texture of SI_FeatureList.

TextureFtrWght_SI
textureFeatureWeight
The texture weight of SI_FeatureList. The default value is 0.0. The weight value can range from 0.0 to 1.0. A value of 0.0 indicates that the feature should not be considered during image matching.

Pragmas

None.

Exceptions

None.

Usage Notes

An error is returned if any of the following conditions is true:

- The AvgClrFtr_SI attribute is not a NULL value and the AvgClrFtrWght_SI attribute value is NULL or less than zero.
- The ClrHstgrFtr_SI attribute is not a NULL value and the ClrHstgrFtrWght_SI attribute value is NULL or less than zero.
- The PstnlClrFtr_SI attribute is not a NULL value and the PstnlClrFtrWght_SI attribute value is NULL or less than zero.
- The TextureFtr_SI attribute is not a NULL value and the TextureFtrWght_SI attribute value is NULL or less than zero.

Examples

Create an SI_FeatureList object using the SI_FeatureList( ) constructor:

```
DECLARE
   myimage       SI_StillImage;
   myAvgColor    SI_AverageColor;
   myColorHist   SI_ColorHistogram;
   myPosColor    SI_PositionalColor;
   myTexture     SI_Texture;
   myFeatureList SI_FeatureList;
BEGIN
   SELECT product_photo INTO myimage FROM pm.si_media WHERE product_id=1 FOR
   UPDATE;
   myAvgColor := NEW SI_AverageColor(myimage);
   myColorHist := NEW SI_ColorHistogram(myimage);
   myPosColor := NEW SI_PositionalColor(myimage);
   myTexture := NEW SI_Texture(myimage);
   myFeatureList := new SI_FeatureList(myAvgColor,0.25,myColorHist,0.35,
                                       myPosColor,0.10, myTexture,0.5);
   UPDATE pm.si_media SET feature_list = myFeatureList WHERE product_id=1;
   COMMIT;
END;
/
```

Create an SI_FeatureList object using the SI_MkFtrList( ) function:

```
DECLARE
   myimage       SI_StillImage;
   myAvgColor    SI_AverageColor;
   myColorHist   SI_ColorHistogram;
```

```
   myPosColor    SI_PositionalColor;
   myTexture     SI_Texture;
   myFeatureList SI_FeatureList;
BEGIN
   SELECT product_photo INTO myimage
     FROM pm.si_media WHERE product_id=2 FOR
    UPDATE;
   myAvgColor := NEW SI_AverageColor(myimage);
   myColorHist := NEW SI_ColorHistogram(myimage);
   myPosColor := NEW SI_PositionalColor(myimage);
   myTexture := NEW SI_Texture(myimage);
   myFeatureList := SI_MkFtrList(myAvgColor,0.25,myColorHist,0.35,
                    myPosColor,0.10, myTexture,0.5);
   UPDATE pm.si_media SET feature_list = myFeatureList WHERE product_id=2;
  COMMIT;
END;
/
```

SI_FeatureList Methods

This section presents reference information on the SI_FeatureList methods used for image matching, which are the following:

SI_AvgClrFtr( )

Format

SI_AvgClrFtr( )

RETURN SI_AverageColor DETERMINISTIC;

Format of Equivalent SQL Function

SI_GetAvgClrFtr(featureList IN SI_FeatureList)

RETURN SI_AverageColor DETERMINISTIC;

Description

Returns the value of the AvgClrFtr_SI attribute of the specified SI_FeatureList object.

Parameters

featureList
The SI_FeatureList object for which you want the AvgClrFtr_SI attribute returned.

Usage Notes

None.

Method Pragmas

PRAGMA RESTRICT_REFERENCES(SI_AvgClrFtr, WNDS, WNPS, RNDS, RNPS)

Function Pragmas

PRAGMA RESTRICT_REFERENCES(SI_GetAvgClrFtr, WNDS, WNPS, RNDS, RNPS)

Exceptions

None.

Examples

Get the value of the AvgClrFtr_SI attribute of an SI_FeatureList object using the SI_AvgClrFtr( ) method:

```
DECLARE
   myFeatureList SI_FeatureList;
   AvgColor    SI_AverageColor;
BEGIN
   SELECT feature_list INTO myFeatureList FROM pm.si_media
     WHERE product_id=1;
   AvgColor := myFeatureList.SI_AvgClrFtr( );
END;
/
```

Get the AvgClrFtr_SI attribute of an SI_FeatureList object using the SI_GetAvgClrFtr( ) function:

```
DECLARE
   myFeatureList SI_FeatureList;
   AvgColor      SI_AverageColor;
BEGIN
   SELECT feature_list INTO myFeatureList FROM pm.si_media WHERE product_id=1;
   AvgColor  := SI_GetAvgClrFtr(myFeatureList);
END;
/
```

SI_AvgClrFtrWght( )

Format

```
SI_AvgClrFtrWght( )
  RETURN DOUBLE PRECISION DETERMINISTIC;
```

Format of Equivalent SQL Function

```
SI_GetAvgClrFtrW(featureList IN SI_FeatureList)
 RETURN DOUBLE PRECISION DETERMINISTIC;
```

Description

Returns the value of the AvgClrFtrWght_SI attribute of the specified SI_FeatureList object.

Parameters

featureList
The SI_FeatureList object for which you want the AvgClrFtrWght_SI attribute returned.

Usage Notes

None.

Method Pragmas

PRAGMA RESTRICT_REFERENCES(SI_AvgClrFtrWght, WNDS, WNPS, RNDS, RNPS)

Function Pragmas

PRAGMA RESTRICT_REFERENCES(SI_GetAvgClrFtrW, WNDS, WNPS, RNDS, RNPS)

Exceptions

None.

Examples

Get the AvgClrFtrWght_SI attribute value of an SI_FeatureList object using the SI_AvgClrFtrWght( ) method:

```
DECLARE
   myFeatureList SI_FeatureList;
   AvgColorWght  DOUBLE PRECISION;
BEGIN
   SELECT feature_list INTO myFeatureList FROM pm.si_media
     WHERE product_id=1;
   AvgColorWght := myFeatureList.SI_AvgClrFtrWght( );
   DBMS_OUTPUT.PUT_LINE('Average color feature weight is ' || AvgColorWght);
END;
/
```

Get the AvgClrFtrWght_SI attribute value of an SI_FeatureList object using the SI_GetAvgClrFtrW( ) function:

```
DECLARE
   myFeatureList SI_FeatureList;
   AvgColorWght  DOUBLE PRECISION;
BEGIN
   SELECT feature_list INTO myFeatureList FROM pm.si_media
     WHERE product_id=1;
   AvgColorWght := SI_GetAvgClrFtrW(myFeatureList);
   DBMS_OUTPUT.PUT_LINE('Average color feature weight is ' || AvgColorWght);
END;
/
```

SI_ClrHstgrFtr( )

Format

SI_ClrHstgrFtr( )

RETURN SI_ColorHistogram DETERMINISTIC;

Format of Equivalent SQL Function

SI_GetClrHstgrFtr(featureList IN SI_FeatureList)

RETURN SI_ColorHistogram DETERMINISTIC;

Description

Returns the value of the ClrHstgrFtr_SI attribute of the specified SI_FeatureList object.

Parameters

featureList
The SI_FeatureList object for which you want the ColorHistogram_SI attribute returned.

Usage Notes

None.

Method Pragmas

PRAGMA RESTRICT_REFERENCES(SI_ClrHstgrFtr, WNDS, WNPS, RNDS, RNPS)

Function Pragmas

PRAGMA RESTRICT_REFERENCES(SI_GetClrHstgrFtr, WNDS, WNPS, RNDS, RNPS)

Exceptions

None.

Examples

Get the value of the ClrHstgrFtr_SI attribute of an SI_FeatureList object using the SI_ClrHstgrFtr( ) method:

```
DECLARE
   myFeatureList SI_FeatureList;
   ClrHstgrFtr   SI_ColorHistogram;
BEGIN
   SELECT feature_list INTO myFeatureList FROM pm.si_media
     WHERE product_id=1;
   ClrHstgrFtr := myFeatureList.SI_ClrHstgrFtr( );
END;
/
```

Get the value of the ClrHstgrFtr_SI attribute of an SI_FeatureList object using the SI_GetClrHstgrFtr( ) function:

```
DECLARE
   myFeatureList SI_FeatureList;
   ClrHstgrFtr   SI_ColorHistogram;
BEGIN
   SELECT feature_list INTO myFeatureList FROM pm.si_media
     WHERE product_id=1;
   ClrHstgrFtr := SI_GetClrHstgrFtr(myFeatureList);
END;
/
```

SI_ClrHstgrFtrWght( )

Format

SI_ClrHstgrFtrWght( )

RETURN DOUBLE PRECISION DETERMINISTIC;

Format of Equivalent SQL Function

SI_GetClrHstgrFtrW(featureList IN SI_FeatureList)

RETURN DOUBLE PRECISION DETERMINISTIC;

Description

Returns the value of the ClrHstgrFtrWght_SI attribute of the specified SI_FeatureList object.

Parameters

featureList
The SI_FeatureList object for which you want the ClrHstgrFtrWght_SI attribute returned.

Usage Notes

None.

Method Pragmas

PRAGMA RESTRICT_REFERENCES(SI_ClrHstgrFtrWght, WNDS, WNPS, RNDS, RNPS)

Function Pragmas

PRAGMA RESTRICT_REFERENCES(SI_GetClrHstgrFtrW, WNDS, WNPS, RNDS, RNPS)

Exceptions

None.

Examples

Get the value of the ClrHstgrFtrWght_SI attribute of an SI_FeatureList object using the SI_ClrHstgrFtrWght( ) method:

```
DECLARE
   myFeatureList    SI_FeatureList;
   ClrHstgrFtrWght  DOUBLE PRECISION;
BEGIN
  SELECT feature_list INTO myFeatureList FROM pm.si_media
    WHERE product_id=1;
  ClrHstgrFtrWght := myFeatureList.SI_ClrHstgrFtrWght( );
  DBMS_OUTPUT.PUT_LINE('Color histogram feature weight is ' || ClrHstgrFtrWght);
END;
/
```

Get the value of the ClrHstgrFtrWght_SI attribute of an SI_FeatureList object using the SI_GetClrHstgrFtrW( ) function:

```
DECLARE
  myFeatureList    SI_FeatureList;
   ClrHstgrFtrWght  DOUBLE PRECISION;
BEGIN
  SELECT feature_list INTO myFeatureList FROM pm.si_media
   WHERE product_id=1;
  ClrHstgrFtrWght := SI_GetClrHstgrFtrW(myFeatureList);
  DBMS_OUTPUT.PUT_LINE('Color histogram feature weight is ' || ClrHstgrFtrWght);
END;
/
```

SI_PstnlClrFtr( )

Format

SI_PstnlClrFtr( )

RETURN SI_PositionalColor DETERMINISTIC;

Format of Equivalent SQL Function

SI_GetPstnlClrFtr(featureList IN SI_FeatureList)

RETURN SI_PositionalColor DETERMINISTIC;

Description

Returns the value of the PstnlClrFtr_SI attribute of the specified SI_FeatureList object.

Parameters

featureList
The SI_FeatureList object for which you want the PstnlClrFtr_SI attribute returned.

Usage Notes

None.

Method Pragmas

PRAGMA RESTRICT_REFERENCES(SI_PstnlClrFtr, WNDS, WNPS, RNDS, RNPS)

Function Pragmas

PRAGMA RESTRICT_REFERENCES(SI_GetPstnlClrFtr, WNDS, WNPS, RNDS, RNPS)

Exceptions

None.

Examples

Get the value of the PstnlClrFtr_SI attribute of an SI_FeatureList object using the SI_PstnlClrFtr( ) method:

```
DECLARE
   myFeatureList SI_FeatureList;
   PstnlClrFtr   SI_PositionalColor;
BEGIN
   SELECT feature_list INTO myFeatureList FROM pm.si_media
     WHERE product_id=1;
   PstnlClrFtr := myFeatureList.SI_PstnlClrFtr( );
END;
/
```

Get the value of the PstnlClrFtr_SI attribute of an SI_FeatureList object using the SI_GetPstnlClrFtr( ) function:

```
DECLARE
   myFeatureList SI_FeatureList;
   PstnlClrFtr   SI_PositionalColor;
BEGIN
   SELECT feature_list INTO myFeatureList FROM pm.si_media WHERE product_id=1;
   PstnlClrFtr := SI_GetPstnlClrFtr(myFeatureList);
END;
/
```

SI_PstnlClrFtrWght( )

Format

SI_PstnlClrFtrWght( )

RETURN DOUBLE PRECISION DETERMINISTIC;

Format of Equivalent SQL Function

SI_GetPstnlClrFtrW(featureList IN SI_FeatureList)

RETURN DOUBLE PRECISION DETERMINISTIC;

Description

Returns the value of the PstnlClrFtrWght_SI attribute of the specified SI_FeatureList object.

Parameters

featureList
The SI_FeatureList object for which you want the PstnlClrFtrWght_SI attribute returned.

Usage Notes

None.

Method Pragmas

PRAGMA RESTRICT_REFERENCES(SI_PstnlClrFtrWght, WNDS, WNPS, RNDS, RNPS)

Function Pragmas

PRAGMA RESTRICT_REFERENCES(SI_GetPstnlClrFtrW, WNDS, WNPS, RNDS, RNPS)

Exceptions

None.

Examples

Get the value of the PstnlClrFtrWght_SI attribute of an SI_FeatureList object using the SI_PstnlClrFtrWght( ) method:

```
DECLARE
   myFeatureList    SI_FeatureList;
   PstnlClrFtrWght  DOUBLE PRECISION;
BEGIN
 SELECT feature_list INTO myFeatureList FROM pm.si_media
   WHERE product_id=1;
 PstnlClrFtrWght := myFeatureList.SI_PstnlClrFtrWght( );
 DBMS_OUTPUT.PUT_LINE('Positional color feature weight is ' || PstnlClrFtrWght);
END;
/
```

Get the value of the PstnlClrFtrWght_SI attribute of an SI_FeatureList object using the SI_GetPstnlClrFtrW( ) function:

```
DECLARE
   myFeatureList    SI_FeatureList;
   PstnlClrFtrWght  DOUBLE PRECISION;
BEGIN
 SELECT feature_list INTO myFeatureList FROM pm.si_media
   WHERE product_id=1;
 PstnlClrFtrWght := SI_GetPstnlClrFtrW(myFeatureList);
 DBMS_OUTPUT.PUT_LINE('Positional color feature weight is ' || PstnlClrFtrWght);
END;
/
```

SI_Score( ) for SI_FeatureList

Format

```
SI_Score(image  IN  SI_StillImage)
   RETURN DOUBLE PRECISION DETERMINISTIC;
```

Format of Equivalent SQL Function

```
SI_ScoreByFtrList(featureList IN SI_FeatureList,
                  image IN SI_StillImage)
RETURN DOUBLE PRECISION DETERMINISTIC;
```

Description

Determines and returns the score of a specified image to a given SI_FeatureList value. The lower the returned score value, the better the image is characterized by the SI_FeatureList object used for scoring the image. The return score value is computed as follows:

Let *n* be the number of non-NULL feature attributes of the FeatureList object to which you are applying the method. For *i* ranging from 1 to n, let f_i be the feature attribute and W_i be the value of the corresponding feature weight. The result is the sum of f_i.SI_Score(image) * W_i divided by the sum of W_i. The process by which the score value is determined can also be described by the following expression:

$$\frac{\sum_{i=1}^{n} f_i.\text{SI\_SCORE(image)} * W_i}{\sum_{i=1}^{n} W_i}$$

A DOUBLE PRECISION value between 0 and 100 is returned. A value of 0 means that the image is identical to the feature list object. A value of 100 means that the image is completely different from the feature list object.

Parameters

featureList
The SI_FeatureList object to which the image will be compared.

image
The image whose features are extracted and compared with the specified SI_FeatureList object to get a score value.

Usage Notes

This method returns a NULL value if any of the following conditions is true:

- The feature list to which this method is applied is a NULL value.
- The value of the specified image is NULL.
- The values of AvgClrFtr_SI, ClrHstgrFtr_SI, PstnlClrFtr_SI, and TextureFtr_SI are all NULL.
- The sum of all the feature weights, AvgClrFtrWght_SI, ClrHstgrFtrWght_SI, PstnlClrFtrWght_SI, and TextureFtrWght_SI is 0.

Pragmas

None.

Exceptions

None.

Examples

Compare an image to an SI_FeatureList value and return the score using the SI_Score( ) method:

```
DECLARE
   myimage       SI_StillImage;
   myFeatureList SI_FeatureList;
   score         DOUBLE PRECISION;
BEGIN
  --Check to see if stored feature list is accurate for the stored image:
  SELECT feature_list INTO myFeatureList FROM pm.si_media
    WHERE product_id=1;
  SELECT product_photo INTO myimage FROM pm.si_media
    WHERE product_id=1;
  -- The score will be 0 if the feature list is accurate for the stored image:
  score := myFeatureList.SI_Score(myimage);
  DBMS_OUTPUT.PUT_LINE('Score is ' || score);
END;
/
```

Compare an image to an SI_FeatureList value and return the score using the SI_ScoreByFtrList( ) function:

```
DECLARE
    score         DOUBLE PRECISION;
    myimage       SI_StillImage;
    myFeatureList SI_PositionalColor;
BEGIN
  SELECT feature_list INTO myFeatureList FROM pm.si_media
    WHERE product_id=1;
  SELECT product_photo INTO myimage FROM pm.si_media
    WHERE product_id=1;
   score := SI_ScoreByFtrList(myFeatureList, myimage);
   DBMS_OUTPUT.PUT_LINE('Score is ' || score);
END;
/
```

SI_SetFeature(averageColorFeature, averageColorFeatureWeight)

Format

```
SI_SetFeature(averageColorFeature        IN  SI_AverageColor,
              averageColorFeatureWeight  IN  DOUBLE PRECISION);
```

Format of Equivalent SQL Procedure

```
SI_SetAvgClrFtr (featureList IN OUT NOCOPY SI_FeatureList,
averageColorFeature          IN SI_AverageColor,
averageColorFeatureWeight IN DOUBLE PRECISION);
```

Description

Modifies the SI_AvgClrFtr and SI_AvgClrFtrWght attributes in the specified SI_FeatureList object.

Parameters

averageColorFeature
The new average color value.

averageColorFeatureWeight
The new average color weight.

featureList
The SI_FeatureList object for which you want to update the averageColorFeature and averageColorFeatureWeight values.

Usage Notes

- If the value of the averageColorFeature parameter is NULL, then the attribute AvgClrFtrWght_SI is set to zero and the value of the averageColorFeatureWeight parameter is disregarded.
- An error is returned if the value of the averageColorFeature parameter is not a NULL value and the corresponding averageColorFeatureWeight parameter value is NULL or less than zero.

Pragmas

None.

Exceptions

None.

Examples

Modify the SI_AvgClrFtr and SI_AvgClrFtrWght attributes in an SI_FeatureList value using the SI_SetFeature(averageColorFeature, averageColorFeatureWeight) method:

```
DECLARE
   myColor       SI_Color;
   myAvgColor    SI_AverageColor;
   myFeatureLIst SI_FeatureList;
BEGIN
   -- Create a new average color:
   myColor := NEW SI_COLOR(null, null, null);
   myColor.SI_RGBColor(30, 120, 200);
   myAvgColor := NEW SI_AverageColor(myColor);
   -- Get an existing feature list:
   SELECT feature_list INTO myFeatureList FROM pm.si_media WHERE product_id=1;
   -- Modify the average color and weight of the SI_FeatureList value:
   myFeatureList.SI_SetFeature(myAvgColor, 0.5);
END;
/
```

Modify the SI_AvgClrFtr and SI_AvgClrFtrWght attributes in an SI_FeatureList object using the SI_SetAvgClrFtr ( ) procedure:

```
DECLARE
   myColor       SI_Color;
   myAvgColor    SI_AverageColor;
   myFeatureLIst SI_FeatureList;
BEGIN
   myColor := NEW SI_COLOR(null, null, null);
   myColor.SI_RGBColor(30, 120, 200);
   myAvgColor := NEW SI_AverageColor(myColor);
   SELECT feature_list INTO myFeatureList FROM pm.si_media WHERE product_id=1;
   SI_SetAvgClrFtr(myFeatureList, myAvgColor, 0.5);
END;
/
```

SI_SetFeature(colorHistogramFeature, colorHistogramFeatureWeight)

Format

```
SI_SetFeature(colorHistogramFeature       IN  SI_ColorHistogram,
              colorHistogramFeatureWeight IN  DOUBLE PRECISION);
```

Format of Equivalent SQL Procedure

```
SI_SetClrHstgrFtr (featureList IN OUT NOCOPY SI_FeatureList,
colorHistogramFeature          IN SI_ColorHistogram,
colorHistogramFeatureWeight IN DOUBLE PRECISION);
```

Description

Modifies the ClrHstgrFtr_SI attribute and ClrHstgrFtrWght_SI attribute in the specified SI_FeatureList object.

Parameters

colorHistogramFeature
The new color histogram value.

colorHistogramFeatureWeight
The new color histogram weight value.

featureList
The SI_FeatureList object for which you want to update the colorHistogram and colorHistogramFeatureWeight attribute values.

Usage Notes

- If the value of the colorHistogramFeature parameter is NULL, then the attribute ClrHstgrFtrWght_SI is set to zero and the value of the colorHistogramFeatureWeight parameter is disregarded.
- An error is returned if the value of the colorHistogramFeature parameter is not a NULL value and the corresponding colorHistogramFeatureWeight parameter value is NULL or less than zero.

Pragmas

None.

Exceptions

None.

Examples

Modify the color histogram feature and feature weight of an SI_FeatureList object using the SI_SetFeature( ) method:

```
DECLARE
   myColor       SI_Color;
   myClrHist   SI_ColorHistogram;
   myFeatureList SI_FeatureList;
BEGIN
   myColor := new SI_Color(null, null, null);
   myColor.SI_RGBColor(10, 45, 200);
   myClrHist := new SI_ColorHistogram(myColor, 10.0);
   SELECT feature_list INTO myFeatureList FROM pm.si_media WHERE product_id=1;
   myFeatureList.SI_SetFeature(myClrHist, 10.5);
END;
/
```

Modify the color histogram feature and feature weight of an SI_FeatureList object using the SI_SetClrHstgrFtr( ) procedure:

```
DECLARE
   myColor       SI_Color;
   myClrHist   SI_ColorHistogram;
   myFeatureList SI_FeatureList;
BEGIN
   myColor := new SI_Color(null, null, null);
   myColor.SI_RGBColor(10, 45, 200);
   myClrHist := new SI_ColorHistogram(myColor, 10.0);
   SELECT feature_list INTO myFeatureList FROM pm.si_media
     WHERE product_id=1;
   SI_SetClrHstgrFtr(myFeatureList, myClrHist, 10.5);
END;
/
```

SI_SetFeature(positionalColorFeature, positionalColorFeatureWeight)

Format

```
SI_SetFeature(positionalColorFeature IN  SI_PositionalColor,
        positionalColorFeatureWeight  IN  DOUBLE PRECISION);
```

Format of Equivalent SQL Procedure

```
SI_SetPstnlClrFtr(featureList  IN OUT NOCOPY SI_FeatureList,
                  positionalColorFeature          IN SI_PositionalColor,
                  positionalColorFeatureWeight IN DOUBLE PRECISION);
```

Description

Modifies the PstnlClrFtr_SI and the PstnlClrFtrWght_SI attributes in the specified SI_FeatureList object.

Parameters

positionalColorFeature
The new positional color value.

positionalColorFeatureWeight
The new positional color weight value.

featureList
The SI_FeatureList object for which you want to update the positionalColor and positionalColorFeatureWeight attributes.

Usage Notes

- If the value of the positionalColorFeature parameter is NULL, the attribute PstnlClrFtrWght_SI is set to zero and the value of the positionalColorFeatureWeight parameter is disregarded.
- An error is returned if the value of the positionalColorFeature parameter is not NULL and the positionalColorFeatureWeight parameter value is NULL or less than zero.

Pragmas

None.

Exceptions

None.

Examples

Modify the positional color feature and feature weight of one image by updating it with the positional color feature value from another image using the SI_SetFeature( ) method and specifying a desired feature weight value:

```
DECLARE
   PosColor2        SI_PositionalColor;
   myFeatureList    SI_FeatureList;
BEGIN
   SELECT positional_color INTO PosColor2  FROM pm.si_media
     WHERE product_id=2;
   SELECT feature_list INTO myFeatureList FROM pm.si_media
     WHERE product_id=1;
   myFeatureList.SI_SetFeature(PosColor2, 10.5);
END;
/
```

Modify the positional color feature and feature weight of one image by updating it with the positional color feature value from another image using the SI_SetPstnlClrFtr( ) procedure and specifying a desired feature weight value:

```
DECLARE
   PosColor1        SI_PositionalColor;
   myFeatureList2   SI_FeatureList;
   myFeatureListnew SI_FeatureList;
BEGIN
   SELECT positional_color INTO PosColor1
     FROM pm.si_media WHERE product_id=1;
   SELECT feature_list INTO myFeatureList2
     FROM pm.si_media WHERE product_id=2;
   SI_SetPstnlClrFtr(myFeatureList2, PosColor1, 10.5);
END;
/
```

SI_SetFeature(textureFeature, textureFeatureWeight)

Format

```
SI_SetFeature(textureFeature        IN  SI_Texture,
              textureFeatureWeight  IN  DOUBLE PRECISION);
```

Format of Equivalent SQL Procedure

```
SI_SetTextureFtr(featureList          IN OUT NOCOPY SI_FeatureList,
                 textureFeature       IN SI_Texture,
                 textureFeatureWeight IN DOUBLE PRECISION);
```

Description

Modifies the TextureFtr_SI attribute and TextureFtrWght_SI attribute in the specified SI_FeatureList object.

Parameters

textureFeature
The new texture value.

textureFeatureWeight
The new texture weight value.

featureList
The SI_FeatureList object for which you want to update the textureFeature and textureFeatureWeight attributes.

Usage Notes

- If the value of the textureFeature parameter is a NULL value and the attribute TextureFtrWght_SI is set to zero, then the value of the textureFeatureWeight parameter is disregarded.
- An error is returned if the value of the textureFeature parameter is NULL and the textureFeatureWeight parameter value is NULL or less than zero.

Pragmas

None.

Exceptions

None.

Examples

Modify the texture feature and feature weight of an SI_FeatureList value using the SI_SetFeature( ) method:

```
DECLARE
   texture1         SI_Texture;
   myFeatureList1   SI_FeatureList;
   myFeatureList2   SI_FeatureList;
BEGIN
   SELECT texture INTO texture1  FROM pm.si_media
     WHERE product_id=1;
   SELECT feature_list INTO myFeatureList1 FROM pm.si_media
     WHERE product_id=2;
   myFeatureList2.SI_SetFeature(texture1, 20);
END;
/
```

Modify the texture feature and feature weight of an SI_FeatureList value using the SI_SetTextureFtr( ) procedure:

```
DECLARE
   texture1         SI_Texture;
   myFeatureList    SI_FeatureList;
BEGIN
   SELECT texture INTO texture1  FROM pm.si_media
     WHERE product_id=1;
   SELECT feature_list INTO myFeatureList FROM pm.si_media
     WHERE product_id=2;
   SI_SetTextureFtr(myFeatureList, texture1, 20);
END;
/
```

SI_TextureFtr( )

Format

```
SI_TextureFtr( )
RETURN SI_Texture DETERMINISTIC;
```

Format of Equivalent SQL Function

```
SI_GetTextureFtr (featureList IN SI_FeatureList)
 RETURN SI_Texture DETERMINISTIC;
```

Description

Returns the value of the TextureFtr_SI attribute of the specified SI_FeatureList object.

Parameters

featureList
The SI_FeatureList object for which you want the TextureFtr_SI attribute returned.

Usage Notes

None.

Method Pragmas

PRAGMA RESTRICT_REFERENCES(SI_TextureFtr, WNDS, WNPS, RNDS, RNPS)

Function Pragmas

PRAGMA RESTRICT_REFERENCES(SI_GetTextureFtr, WNDS, WNPS, RNDS, RNPS)

Exceptions

None.

Examples

Get the value of the TextureFtr_SI attribute of an SI_FeatureList object using the SI_TextureFtr( ) method:

```
DECLARE
   myFeatureList SI_FeatureList;
   mytexture     SI_Texture;
BEGIN
   SELECT feature_list INTO myFeatureList FROM pm.si_media
     WHERE product_id=1;
   mytexture := myFeatureList.SI_TextureFtr( );
END;
/
```

Get the value of the TextureFtr_SI attribute of an SI_FeatureList object using the SI_GetTextureFtr ( ) function:

```
DECLARE
   myFeatureList SI_FeatureList;
   mytexture       SI_Texture;
BEGIN
   SELECT feature_list INTO myFeatureList FROM pm.si_media WHERE product_id=1;
   mytexture := SI_GetTextureFtr(myFeatureList);
END;
/
```

SI_TextureFtrWght( )

Format

SI_TextureFtrWght( )

RETURN DOUBLE PRECISION DETERMINISTIC;

Format of Equivalent SQL Function

SI_GetTextureFtrW(featureList in SI_FeatureList)

RETURN DOUBLE PRECISION DETERMINISTIC;

Description

Returns the value of the TextureFtrWght_SI attribute of the specified SI_FeatureList object.

Parameters

featureList
The SI_FeatureList object for which you want the TextureFtrWght_SI attribute returned.

Usage Notes

None.

Method Pragmas

PRAGMA RESTRICT_REFERENCES(SI_TextureFtrWght, WNDS, WNPS, RNDS, RNPS)

Function Pragmas

PRAGMA RESTRICT_REFERENCES(SI_GetTextureFtrW, WNDS, WNPS, RNDS, RNPS)

Exceptions

None.

Examples

Get the TextureFtrWght_SI attribute of an SI_FeatureList object using the SI_TextureFtrWght( ) method:

```
DECLARE
   myFeatureList SI_FeatureList;
   myTextureWght DOUBLE PRECISION;
BEGIN
   SELECT feature_list INTO myFeatureList FROM pm.si_media
     WHERE product_id=1;
   myTextureWght := myFeatureList.SI_TextureFtrWght( );
   DBMS_OUTPUT.PUT_LINE('Texture feature weight is ' || myTextureWght);
END;
/
```

Get the value of the TextureFtrWght_SI attribute of an SI_FeatureList object using the SI_GetTextureFtrW( ) function:

```
DECLARE
   myFeatureList SI_FeatureList;
   myTextureWght DOUBLE PRECISION;
BEGIN
   SELECT feature_list INTO myFeatureList FROM pm.si_media
     WHERE product_id=1;
   myTextureWght := SI_GetTextureFtrW(myFeatureList);
   DBMS_OUTPUT.PUT_LINE('Texture feature weight is ' || myTextureWght);
END;
/
```

SI_PositionalColor Object Type

The SI_PositionalColor object represents the most significant color positions of an image. If an image is divided into *n* by *m* rectangles, positional color is a feature that characterizes the image by the *n* by *m* most significant colors of the rectangles. This object type is created in the ORDSYS schema with invoker rights. It is declared as an INSTANTIABLE and NOT FINAL type. (See "Internal Helper Types" on page 7-133 for the colorPositions attribute syntax.)

> **Note:** Use the SI_PositionalColor object constructor and method rather than accessing attributes directly to protect yourself from changes to the internal representation of the SI_PositionalColor object.

The SI_PositionalColor object is defined as follows:

```
CREATE OR REPLACE TYPE SI_PositionalColor
  AUTHID CURRENT_USER
  AS OBJECT
  (
     --attributes
     SI_ColorPositions  colorPositions,
     --
     --Methods
     CONSTRUCTOR FUNCTION SI_PositionalColor
     (sourceImage IN SI_StillImage)
     RETURN SELF AS RESULT DETERMINISTIC,
     --
     MEMBER FUNCTION SI_Score
      (image IN SI_StillImage),
     RETURN DOUBLE PRECISION DETERMINISTIC
) INSTANTIABLE
  NOT FINAL;
/
```

where:

- SI_ColorPositions: an array of SI_Color that represents the most significant color positions of an image.

SI_PositionalColor Constructor

This section describes the SI_PositionalColor object constructor, which is as follows:

SI_PositionalColor( )

Format

```
SI_PositionalColor(sourceImage IN SI_StillImage)
   RETURN SELF AS RESULT DETERMINISTIC;
```

Format of Equivalent SQL Function

```
SI_FindPstnlClr(sourceImage  IN SI_StillImage)
 RETURN SI_PositionalColor DETERMINISTIC;
```

Description

Constructs an SI_PositionalColor object from a specified image. The SI_ColorPositions array attribute is initialized with the most significant color values derived from the specified image.

To derive the SI_PositionalColor object, the image is assumed to be divided into *n* by *m* rectangles such that the product of *n* by *m* is equal to the value of SI_NumberSections. (Query the SI_VALUES view in SI_INFORMTN_SCHEMA for the value of SI_NumberSections.) The most significant color of each rectangle is determined. The array thus computed is the value of the SI_ColorPositions array attribute.

Parameters

sourceImage
Image whose positional color feature is extracted.

Pragmas

None.

Exceptions

None.

Usage Notes

An error is returned if any of the following conditions is true:

- The value of the sourceImage parameter is NULL.

- The value of sourceImage.SI_Content is NULL.
- The positional color feature is not supported for this image format.

You can determine whether or not the positional color feature is supported for an image format by querying the SI_IMAGE_FORMAT_FEATURES view or the SI_IMAGE_FRMT_FTRS view.

Examples

Create an SI_PositionalColor object from an image using the SI_PositionalColor( ) method:

```
DECLARE
   myPosColor SI_PositionalColor;
   myimage    SI_StillImage;
BEGIN
   SELECT product_photo INTO myimage FROM PM.SI_MEDIA
      WHERE product_id=1 FOR UPDATE;
   myPosColor := NEW SI_PositionalColor(myimage);
   UPDATE PM.SI_MEDIA SET positional_color = myPosColor WHERE product_id=1;
   SELECT product_photo INTO myimage FROM PM.SI_MEDIA
      WHERE product_id=2 FOR UPDATE;
   myPosColor := NEW SI_PositionalColor(myimage);
   UPDATE PM.SI_MEDIA SET positional_color = myPosColor WHERE product_id=2;
   COMMIT;
END;
/
```

Create an SI_PositionalColor object from an image using the SI_FindPstnlClr( ) function:

```
DECLARE
   myPosColor SI_PositionalColor;
   myimage    SI_StillImage;
BEGIN
   SELECT product_photo INTO myimage FROM PM.SI_MEDIA
     WHERE product_id=2 FOR UPDATE;
   UPDATE PM.SI_MEDIA SET positional_color = SI_FindPstnlClr(myimage)
     WHERE product_id=2;
   COMMIT;
END;
/
```

SI_PositionalColor Method

This section presents reference information on the SI_PositionalColor method used for image matching, which is as follows:

SI_Score( ) for SI_PositionalColor

Format

```
SI_Score(image IN SI_StillImage)
  RETURN DOUBLE PRECISION DETERMINISTIC;
```

Format of Equivalent SQL Function

```
SI_ScoreByPstnlClr(feature IN SI_PositionalColor,
                   image IN SI_StillImage),
RETURN DOUBLE PRECISION DETERMINISTIC;
```

Description

Determines and returns the score of the specified image when compared to the SI_PositionalColor object to which this method is applied. For scoring an image, that image is divided into *n* by *m* rectangles such that the product (*m* * *n*) is equal to SI_NumberSections. (Query the SI_VALUES view in SI_INFORMTN_SCHEMA for the value of SI_NumberSections.) The lower the returned value, the better the *n* by *m* most significant colors of the image are characterized by the most significant colors in SI_PositionalColor to which you apply this method.

This method returns a DOUBLE PRECISION value between 0 and 100, unless any one of the following is true, in which case a NULL value is returned:

- The value of the SI_PositionalColor object to which you apply this method is NULL.
- The value of the image parameter is NULL.
- The value of image.content_SI attribute is NULL.
- The positional color feature is not supported for the specified image.

Parameters

feature
The positional color to be compared with the positional color of the specified image.

image
The image whose positional color feature is extracted and used for comparison.

Usage Notes

None.

Pragmas

None.

Exceptions

None.

Examples

Compare an image to an SI_PositionalColor object and return the score using the SI_Score( ) method:

```
DECLARE
    score DOUBLE PRECISION;
    myimage SI_StillImage;
    myPosColor SI_PositionalColor;
BEGIN
   SELECT positional_color INTO myPosColor FROM PM.SI_MEDIA
     WHERE product_id=1;
   SELECT product_photo INTO myimage FROM PM.SI_MEDIA WHERE product_id=2;
   score := myPosColor.SI_Score(myimage);
   DBMS_OUTPUT.PUT_LINE('Score is ' || score);
END;
/
```

Compare an image to an SI_PositionalColor object and return the score using the SI_ScoreByPstnlClr( ) function:

```
DECLARE
    score DOUBLE PRECISION;
    myimage SI_StillImage;
    myPosColor SI_PositionalColor;
BEGIN
   SELECT positional_color INTO myPosColor FROM PM.SI_MEDIA
      WHERE product_id=1;
   SELECT product_photo INTO myimage FROM PM.SI_MEDIA WHERE product_id=2;
   score := SI_ScoreByPstnlClr(myPosColor, myimage);
   DBMS_OUTPUT.PUT_LINE('Score is ' || score);
END;
/
```

SI_StillImage Object Type

The SI_StillImage object type represents digital images with inherent image characteristics such as height, width, format, and so on. It is created in the ORDSYS schema with invoker rights and it is declared as INSTANTIABLE and NOT FINAL.

> **Note:** Use the SI_StillImage constructors and methods rather than accessing attributes directly to protect yourself from changes to the internal representation of the SI_StillImage object.

This object type is defined as follows:

```
CREATE OR REPLACE TYPE SI_StillImage
AUTHID CURRENT_USER
AS OBJECT
(
-------------------
-- TYPE ATTRIBUTES
-------------------
content_SI           ORDSYS.ORDSOURCE,
contentLength_SI     INTEGER,
format_SI            VARCHAR2(4000),
height_SI            INTEGER,
width_SI             INTEGER,
-- Oracle attribute extensions
mimeType_ora          VARCHAR2(4000),
contentFormat_ora     VARCHAR2(4000),
compressionFormat_ora VARCHAR2(4000),
-- Flag to
retainFeatures_SI     INTEGER,
-- Oracle extension attributes to cache image features
 averageColorSpec_ora SI_Color,
 colorsList_ora       colorsList,
 frequenciesList_ora  colorFrequenciesList,
 colorPositions_ora   colorPositions,
 textureEncoding_ora  textureEncoding,
---------------------
-- METHOD DECLARATION
---------------------
-- CONSTRUCTORS
--
```

```
CONSTRUCTOR FUNCTION
SI_StillImage(content IN BLOB) RETURN SELF as RESULT DETERMINISTIC,
CONSTRUCTOR FUNCTION
SI_StillImage(content IN BLOB,
              explicitFormat IN VARCHAR2
              ) RETURN SELF AS RESULT DETERMINISTIC,
CONSTRUCTOR FUNCTION
SI_StillImage(content IN BLOB,
              explicitFormat IN VARCHAR2,
              height IN INTEGER,
              width IN INTEGER)  RETURN SELF as RESULT DETERMINISTIC,

-- Accessor methods for StillImage attributes
MEMBER FUNCTION SI_Height RETURN INTEGER DETERMINISTIC,
PRAGMA RESTRICT_REFERENCES(SI_Height, WNDS, WNPS, RNDS, RNPS),

MEMBER FUNCTION SI_Width RETURN INTEGER DETERMINISTIC,
PRAGMA RESTRICT_REFERENCES(SI_Width, WNDS, WNPS, RNDS, RNPS),

MEMBER FUNCTION SI_Format RETURN VARCHAR2 DETERMINISTIC,
PRAGMA RESTRICT_REFERENCES(SI_Format, WNDS, WNPS, RNDS, RNPS),

MEMBER FUNCTION SI_Content RETURN BLOB DETERMINISTIC,
PRAGMA RESTRICT_REFERENCES(SI_Content, WNDS, WNPS, RNDS, RNPS),

MEMBER FUNCTION SI_ContentLength RETURN INTEGER DETERMINISTIC,
PRAGMA RESTRICT_REFERENCES(SI_ContentLength, WNDS, WNPS, RNDS, RNPS),

--Accessor method for retainFeatures_SI attribute
MEMBER FUNCTION SI_retainFeatures return BOOLEAN DETERMINISTIC,
PRAGMA RESTRICT_REFERENCES(SI_retainFeatures, WNDS, WNPS, RNDS, RNPS),
-- Methods associated with image processing
MEMBER PROCEDURE SI_SetContent(content IN BLOB),
MEMBER PROCEDURE SI_ChangeFormat(targetFormat IN VARCHAR2),
MEMBER FUNCTION SI_Thumbnail( )
                    return SI_StillImage DETERMINISTIC,
MEMBER FUNCTION SI_Thumbnail(height IN INTEGER,
                             width IN INTEGER)
                    return SI_StillImage DETERMINISTIC,
-- Methods associated with the Oracle extension for image feature caching
MEMBER PROCEDURE   SI_InitFeatures,
MEMBER PROCEDURE   SI_ClearFeatures
) INSTANTIABLE
NOT FINAL;
/
```

where:

- content_SI: an ORDSource object that contains the binary image or BLOB. (SQL/MM specifies the SI_Content attribute as a BLOB.)
- contentLength_SI: the content length of the image in bytes.
- format_SI: the image format.
- height_SI: the number of lines of the image.
- width_SI: the number of columns of the image.
- mimeType_ora: the MIME type information. (This is an Oracle extension to the SQL/MM Still Image standard.)
- contentFormat_ora: the type of image (monochrome and so on). (This is an Oracle extension to the SQL/MM Still Image standard.)
- compressionFormat_ora: the compression algorithm used on the image data. (This is an Oracle extension to the SQL/MM Still Image standard.)
- retainFeatures_SI: a flag that indicates whether or not image features will be extracted and cached.
- averageColorSpec_ora: the cached SI_Color object.
- colorsList_ora: the cached array of colors.
- frequenciesList_ora: the cached array of color frequencies.
- colorPositions_ora: the cached array of color positions.
- textureEncoding_ora: the cached array of textures.

SI_StillImage Constructors

This section describes the SI_StillImage object constructors, which are the following:

- SI_StillImage(content) on page 7-83
- SI_StillImage(content, explicitFormat) on page 7-87
- SI_StillImage(content, explicitFormat, height, width) on page 7-91

  This is an Oracle extension to the SQL/MM Still Image standard.

> **Note:** To construct SI_StillImage objects, Oracle Corporation strongly recommends that you use one of the constructors in the previous list, not the default SI_StillImage object constructor.

SI_StillImage(content)

Format

```
SI_StillImage(content IN BLOB)
 RETURN SELF as RESULT DETERMINISTIC;
```

Format of Equivalent SQL Function

```
SI_MkStillImage1(content in BLOB)
 RETURN SI_StillImage DETERMINISTIC;
```

Description

Returns a new SI_StillImage object. This constructor initializes the SI_StillImage attributes as follows:

- content_SI.localData is initialized with the specified image.
- contentLength_SI is initialized with the length of the image extracted from the specified image.
- format_SI is initialized with the format of image extracted from the specified image.
- height_SI is initialized with the height of image extracted from the specified image.
- width_SI is initialized with the width of image extracted from the specified image.

Parameters

content
The image data.

Pragmas

None.

Exceptions

ORDImageSIExceptions.NULL_CONTENT

This exception is raised if the content parameter is NULL.

See Appendix F for more information about these exceptions.

Usage Notes

None.

Examples

The following examples demonstrate how to insert a StillImage object into a database table. The first two examples use the SI_StillImage(content) constructor; the third example uses the SI_MkStillImage1( ) function. In addition, the first and third examples use the PL/SQL package DBMS_LOB.LOADFROM FILE and the second example uses SQL*Loader. Typically, you would use the PL/SQL package if you were inserting objects one-by-one, and you would use SQL*Loader to insert objects in a batch job.

Example 1: Insert a BLOB into a StillImage object column using DBMS_ LOB.LOADFROMFILE.

This example demonstrates how to insert an image into a StillImage object column using the PL/SQL package DBMS_LOB.LOADFROMFILE.

Connect as SYSDBA, create a BFILE to indicate where the image you want to load is located, and grant the READ privilege to the user who will be loading the image:

```
DECLARE
   lobd blob;
   fils BFILE := BFILENAME('FILE_DIR','speaker.jpg');
BEGIN
   DBMS_LOB.CREATETEMPORARY(lobd, TRUE);
   DBMS_LOB.fileopen(fils, DBMS_LOB.file_readonly);
   DBMS_LOB.LOADFROMFILE(lobd, fils, DBMS_LOB.GETLENGTH(fils));
   DBMS_LOB.FILECLOSE(fils);
   INSERT INTO PM.SI_MEDIA (product_id, product_photo)
       VALUES(1, new ORDSYS.SI_StillImage(lobd));
   DBMS_LOB.FREETEMPORARY(lobd);
   COMMIT;
END;
/
```

Example 2: Insert a BLOB into a StillImage object column using SQL*Loader.

This example demonstrates how to insert a StillImage object into a database table using SQL*Loader. This example assumes that the image file, speaker.jpg, is in the same directory as the parameter file and the control file.

1. From the command prompt, run SQL*Loader:

```
sqlldr parfile= blob_load.par
```

2. When the phrase 'commit point reached' is returned, connect to the database as user ron and issue a SQL COMMIT statement. (This example assumes that user ron has been created and granted privileges as shown in Example 1.)

```
sqlplus
CONNECT ron/ron
```

3. Run the SQL script to insert the SI_StillImage object in the SI_MEDIA table:

```
@insert_object.sql
```

This example assumes that the blob_load.par, blob_load.ctl, and insert_object.sql files have been defined as follow and that the PM.IMAGETAB table has been created as shown in Example Media Table and User Definition on page 7-4.

- blob_load.par:

```
userid=ron/ron
control=/homedir/user3/blob_load.ctl
log=/homedir/user3/blob_load.log
```

- blob_load.ctl:

```
LOAD DATA
INFILE *
INTO TABLE PM.IMAGETAB
REPLACE
FIELDS TERMINATED BY ','
(id ,
 imgblob_fname FILLER CHAR(20),
 imgblob LOBFILE(imgblob_fname) raw terminated by EOF)
BEGINDATA
1,speakers.jpg
```

- insert_object.sql:

```
DECLARE
   myblob BLOB;
   myid NUMBER;
BEGIN
   SELECT id, imgblob into myid, myblob FROM PM.IMAGETAB WHERE id = 2;
   INSERT INTO PM.SI_MEDIA (product_id, product_photo)
   VALUES(myid, new ordsys.SI_StillImage(myblob));
```

```
      COMMIT;
   END;
   /
```

Example 3: Create a new SI_StillImage object using the SI_MkStillImage1( ) function and the PL/SQL package DBMS_LOB.LOADFROM FILE:

```
DECLARE
   lobd blob;
   fils BFILE := BFILENAME('FILE_DIR','modem.jpg');
BEGIN
   DBMS_LOB.CREATETEMPORARY(lobd, TRUE);
   DBMS_LOB.FILEOPEN(fils, dbms_lob.file_readonly);
   DBMS_LOB.LOADFROMFILE(lobd, fils, dbms_lob.getlength(fils));
   DBMS_LOB.FILECLOSE(fils);
   INSERT INTO PM.SI_MEDIA (product_id, product_photo)
     VALUES (66, SI_MkStillImage1(lobd));
   DBMS_LOB.FREETEMPORARY(lobd);
   COMMIT;
END;
/
```

SI_StillImage(content, explicitFormat)

Format

```
SI_StillImage(content          IN BLOB,
              explicitFormat   IN VARCHAR2)
 RETURN SELF as RESULT DETERMINISTIC;
```

Format of Equivalent SQL Function

```
SI_MkStillImage2(content in BLOB, explicitFormat in VARCHAR2)
 RETURN SI_StillImage DETERMINISTIC;
```

Description

Constructs an SI_StillImage object from a specified image and a format. This constructor lets you specify the image format when the specified image is in an unsupported image format. Query the SI_IMAGE_FORMATS view in SI_INFORMTN_SCHEMA for a list of the supported image formats.

This constructor initializes the SI_StillImage attributes as follows:

- content_SI.localData is initialized with the specified image.
- contentLength_SI is initialized with the length of the image extracted from the specified image.
- format_SI is initialized with the specified image format.
- height_SI is initialized with the height of the image extracted from the specified image. If the constructor function is not able to extract the height value from the specified image, then you can assign a height value to the height_SI attribute -- for example: `myImage.height_SI := height`.
- width_SI is initialized with the width of the image extracted from the specified image. If the constructor function is not able to extract the width value from the specified image, then you can assign a width value to the width_SI attribute -- for example: `myImage.width_SI := width`.

Parameters

content
The image data.

explicitFormat
The format that you want *inter*Media to use if the specified image is in an unsupported image format.

Pragmas

None.

Exceptions

ORDImageSIExceptions.NULL_CONTENT

This exception is raised if the content parameter is NULL.

Usage Notes

An error is returned if the explicitFormat parameter is a NULL value, or if either of the following statements is true:

- The explicitFormat parameter value is a supported format, but it is not equivalent to the format extracted from the specified image.
- The explicitFormat parameter value is an unsupported format, but the format extracted from the specified image is not a NULL value.

The following table presents values for the explicitFormat parameter and the actual image format, and whether or not that combination of values will result in an error. A image format of NULL indicates that the format cannot be extracted from the image.

explicitFormat	Image Format	Error Returned?
GIF (a supported format)	GIF	No
GIF (a supported format)	JPEG	Yes
xyz (an unsupported format)	GIF	Yes
xyz (an unsupported format)	Null	No

Examples

Create an SI_StillImage object from a specified image and format using the SI_StillImage(content, explicitFormat) constructor:

```
DECLARE
   lobd BLOB;
```

```
   fils BFILE := BFILENAME('FILE_DIR','window.psp');
   newimage SI_StillImage;
   height NUMBER;
   width NUMBER;
   myimage SI_StillImage;
BEGIN
   -- Put the blob in a temporary LOB:
   DBMS_LOB.CREATETEMPORARY(lobd, TRUE);
   DBMS_LOB.FILEOPEN(fils, DBMS_LOB.FILE_READONLY);
   DBMS_LOB.LOADFROMFILE(lobd, fils, DBMS_LOB.GETLENGTH(fils));
   DBMS_LOB.FILECLOSE(fils);
   -- Create a new SI_StillImage object for this image (which has an
   -- unsupported format):
   newimage := NEW SI_StillImage(lobd, 'psp');
   -- If the stored height and width values are NULL, the following will set
   -- them appropriately. Alternatively, you could use the
   -- SI_StillImage(content, explicitFormat, height,width) constructor:
   height := 570;
   width := 1168;
   IF (newimage.SI_Height is NULL) THEN
      newimage.height_SI := height;
   END IF;
   IF (newimage.SI_Width is NULL) THEN
      newimage.width_SI := width;
   END IF;
   -- Insert the image into the si_media table, then free the temporary LOB:
   INSERT INTO PM.SI_MEDIA (product_id, product_photo) VALUES (3, newimage);
   DBMS_LOB.FREETEMPORARY(lobd);
   -- Make sure that the height and width were stored as expected:
   SELECT product_photo INTO myimage FROM PM.SI_MEDIA WHERE product_id=3;
   height := myimage.SI_height;
   width := myimage.SI_width;
   DBMS_OUTPUT.PUT_LINE('Height is ' || height || ' pixels.');
   DBMS_OUTPUT.PUT_LINE('Width is ' || width ||  ' pixels.');
   COMMIT;
END;
/
```

Create a new SI_StillImage object from a specified image and format using the SI_MkStillImage2( ) function:

```
DECLARE
   lobd BLOB;
   fils BFILE := BFILENAME('FILE_DIR','window.psp');
   newimage SI_StillImage;
```

```
   height NUMBER;
   width NUMBER;
   myimage SI_StillImage;
BEGIN
   -- Put the blob in a temporary LOB:
   DBMS_LOB.CREATETEMPORARY(lobd, TRUE);
   DBMS_LOB.FILEOPEN(fils, DBMS_LOB.FILE_READONLY);
   DBMS_LOB.LOADFROMFILE(lobd, fils, DBMS_LOB.GETLENGTH(fils));
   DBMS_LOB.FILECLOSE(fils);
   -- Create a new SI_StillImage object for this image (which has an
   -- unsupported format):
   newimage := SI_MkStillImage2(lobd, 'psp');
   -- If the stored height and width values are NULL, the following will set
   -- them appropriately:
   height := 570;
   width := 1168;
   IF (SI_GetHeight(newimage) is NULL) THEN
      newimage.height_SI := height;
   END IF;
   IF (SI_GetWidth(newimage) is NULL) THEN
      newimage.width_SI := width;
   END IF;
   -- Insert the image into the si_media table, then free the temporary LOB:
   INSERT INTO PM.SI_MEDIA (product_id, product_photo) VALUES (77, newimage);
   DBMS_LOB.FREETEMPORARY(lobd);
   -- Make sure that the height and width were stored as expected:
   SELECT product_photo INTO myimage FROM PM.SI_MEDIA WHERE product_id=77;
   height := myimage.SI_height;
   width := myimage.SI_width;
   DBMS_OUTPUT.PUT_LINE('Height is ' || height || ' pixels.');
   DBMS_OUTPUT.PUT_LINE('Width is ' || width ||  ' pixels.');
   COMMIT;
END;
/
```

SI_StillImage(content, explicitFormat, height, width)

Format

```
SI_StillImage(content  IN BLOB,
        explicitFormat IN VARCHAR2,
                height IN INTEGER,
                 width IN INTEGER)
RETURN SI_STILLIMAGE as RESULT DETERMINISTIC;
```

Format of Equivalent SQL Function

```
ora_SI_MkStillImage(content IN BLOB)
                explicitFormat IN VARCHAR2,
                        height IN INTEGER,
                         width IN INTEGER)
 RETURN SI_StillImage DETERMINISTIC;
```

Description

Constructs an SI_StillImage value from a specified image. This constructor lets you specify the image format, height, and width when the specified image is an unsupported image format. Query the SI_IMAGE_FORMATS view in SI_INFORMTN_SCHEMA for a list of the supported image formats.

This constructor and its equivalent SQL function are Oracle extensions to the SQL/MM Still Image standard.

This constructor initializes the SI_StillImage attributes as follows:

- content_SI.localData is initialized with the specified image.
- contentLength_SI is initialized with the length of the image extracted from the specified image.
- format_SI is initialized with the specified format.
- height_SI is initialized with the specified height if the height cannot be extracted from the specified image.
- width_SI is initialized with the specified width if the width cannot be extracted from the specified image.

Parameters

content
The image data.

explicitFormat
The format that you want *inter*Media to use if the image is in an unsupported format.

height
The value for the height_SI attribute that you want *inter*Media to use if the image is in an unsupported format.

width
The value for the width_SI attribute that you want *inter*Media to use if the image is in an unsupported format.

Pragmas

None.

Exceptions

ORDImageSIExceptions.NULL_CONTENT

This exception is raised if the content parameter is NULL.

ORDImageSIExceptions.ILLEGAL_HEIGHT_WIDTH_SPEC

This exception is raised if the value of the height or width parameter is NULL or is a negative value.

Usage Notes

An error message is returned if the explicitFormat parameter value is a NULL value, or if either of the following statements is true:

- The explicitFormat parameter value is a supported format, but it is not equivalent to the format extracted from the image.
- The explicitFormat parameter value is an unsupported format, but the format extracted from the image is not a NULL value.

The following table presents values for the explicitFormat parameter and the actual image format, and whether or not that combination of values will result in an error.

An image format of NULL indicates that the format cannot be extracted from the image.

explicitFormat	Image Format	Error Returned?
GIF (a supported format)	GIF	No
GIF (a supported format)	JPEG	Yes
xyz (an unsupported format)	GIF	Yes
xyz (an unsupported format)	Null	No

Examples

Construct an SI_StillImage value from an image using the SI_StillImage(content, explicitFormat, height, width) constructor:

```
DECLARE
   lobd BLOB;
   fils BFILE := BFILENAME('FILE_DIR','window.psp');
   newimage SI_StillImage;
   height NUMBER;
   width NUMBER;
   myimage SI_StillImage;
BEGIN
   -- Put the blob in a temporary LOB:
   DBMS_LOB.CREATETEMPORARY(lobd, TRUE);
   DBMS_LOB.FILEOPEN(fils, DBMS_LOB.FILE_READONLY);
   DBMS_LOB.LOADFROMFILE(lobd, fils, DBMS_LOB.GETLENGTH(fils));
   DBMS_LOB.FILECLOSE(fils);
   -- Create a new SI_StillImage object for this image (which has an
   -- unsupported format):
   newimage := NEW SI_StillImage(lobd, 'psp', 570, 1168);
   -- Insert the image into the si_media table, then free the temporary LOB:
   INSERT INTO PM.SI_MEDIA (product_id, product_photo) VALUES (4, newimage);
   DBMS_LOB.FREETEMPORARY(lobd);
   -- Make sure that the height and width were stored as expected:
   SELECT product_photo INTO myimage FROM PM.SI_MEDIA WHERE product_id=4;
   height := myimage.SI_height;
   width := myimage.SI_width;
   DBMS_OUTPUT.PUT_LINE('Height is ' || height || ' pixels.');
   DBMS_OUTPUT.PUT_LINE('Width is ' || width ||  ' pixels.');
   COMMIT;
END;
/
```

Construct an SI_StillImage value from an image using the ora_SI_StillImage(content, explicitFormat, height, width) function:

```
DECLARE
   lobd BLOB;
   fils BFILE := BFILENAME('FILE_DIR','window.psp');
   newimage SI_StillImage;
   height NUMBER;
   width NUMBER;
   myimage SI_StillImage;
BEGIN
   -- Put the blob in a temporary LOB:
   DBMS_LOB.CREATETEMPORARY(lobd, TRUE);
   DBMS_LOB.FILEOPEN(fils, DBMS_LOB.FILE_READONLY);
   DBMS_LOB.LOADFROMFILE(lobd, fils, DBMS_LOB.GETLENGTH(fils));
   DBMS_LOB.FILECLOSE(fils);
   -- Create a new SI_StillImage object for this image (which has an
   -- unsupported format):
   newimage := ora_SI_StillImage(lobd, 'psp', 570, 1168);
   -- Insert the image into the si_media table, then free the temporary LOB:
   INSERT INTO PM.SI_MEDIA (product_id, product_photo) VALUES (6, newimage);
   DBMS_LOB.FREETEMPORARY(lobd);
   -- Make sure that the height and width were stored as expected:
   SELECT product_photo INTO myimage FROM PM.SI_MEDIA WHERE product_id=6;
   height := myimage.SI_height;
   width := myimage.SI_width;
   DBMS_OUTPUT.PUT_LINE('Height is ' || height || ' pixels.');
   DBMS_OUTPUT.PUT_LINE('Width is ' || width ||  ' pixels.');
   COMMIT;
END;
/
```

SI_StillImage Methods

This section presents reference information on the SI_StillImage methods used for image data manipulation, which are the following:

SI_ClearFeatures( )

Format

SI_ClearFeatures( );

Description

Disables image feature caching and sets the value of all internal image feature attributes to NULL. You can call this method to remove the processing overhead associated with feature synchronization if you are not performing image matching. This method does nothing for unsupported image formats.

This method is not in the first edition of the SQL/MM Still Image standard, but has been accepted for inclusion in the next version.

Parameters

None.

Usage Notes

None.

Pragmas

None

Exceptions

None.

Examples

Disable image feature caching and set the value of all internal image feature attributes for a specified image to NULL:

```
DECLARE
  myimage SI_StillImage;
BEGIN
  SELECT product_photo INTO myimage FROM PM.SI_MEDIA WHERE product_id = 1;
  UPDATE PM.SI_MEDIA
   SET product_photo = myimage where product_id=1;
  myimage.SI_ClearFeatures;
  DBMS_OUTPUT.PUT_LINE('Image feature caching diasbled');
  COMMIT;
```

```
END;
/
```

SI_InitFeatures( )

Format

```
SI_InitFeatures( );
```

Description

Extracts the image features and caches them in the SI_StillImage object. This method needs to be called once, after which SI_StillImage will manage the image features such that every time the image is processed, new image features will automatically be extracted. This method is recommended for image-matching users.

This method is not in the first edition of the SQL/MM Still Image standard, but has been accepted for inclusion in the next version.

Parameters

None.

Usage Notes

- The performance impacts associated with image feature caching are:
  - Image processing methods such as SI_SetContent and SI_ChangeFormat will be slower.
  - Image matching methods such as SI_Score will be faster.
- Image feature extraction and caching are not available for unsupported image formats.

Pragmas

None.

Exceptions

ORDImageSIExceptions.UNSUPPORTED_IMAGE_FORMAT

This exception is raised if this method is invoked on an unsupported image format.

Examples

Extract the image features and cache them in an SI_StillImage object:

```
DECLARE
```

```
  myimage SI_StillImage;
BEGIN
  SELECT product_photo INTO myimage FROM PM.SI_MEDIA
     WHERE product_id = 1 FOR UPDATE;
  myimage.SI_InitFeatures;
  UPDATE PM.SI_MEDIA
   SET product_photo = myimage where product_id=1;
  DBMS_OUTPUT.PUT_LINE('Image feature caching enabled');
  COMMIT;
END;
/
```

SI_ChangeFormat( )

Format

```
SI_ChangeFormat(targetFormat IN VARCHAR2);
```

Format of Equivalent SQL Procedure

```
SI_ConvertFormat(image        IN OUT NOCOPY SI_StillImage,
                 targetFormat IN VARCHAR2);
```

Description

Converts the format of an SI_StillImage object and adjusts the affected attributes as follows:

- content_SI is converted to the value specified with the targetFormat parameter.
- contentLength_SI is updated with the new image length extracted from the content_SI attribute.
- format_SI is set equal to the targetFormat parameter value.
- height_SI is updated with the new height extracted from the content_SI attribute.
- width_SI is updated with the new width extracted from the content_SI attribute.

Parameters

image
The image whose content you want to convert.

targetFormat
The format to which you want the image to be converted.

Usage Notes

An error message is returned if any of the following is true:

- The value of the format_SI attribute is NULL.
- The value of the targetFormat parameter is NULL.

- The conversion from format_SI to targetFormat is not supported. (Oracle *inter*Media determines this by looking up the values in the SI_IMAGE_FORMAT_CONVERSIONS view or the SI_FORMAT_CONVRSNS view in SI_INFORMTN_SCHEMA.)

Pragmas

None.

Exceptions

None.

Examples

Convert the format of an SI_StillImage object to WBMP using the SI_ChangeFormat( ) method:

```
DECLARE
   origformat VARCHAR2 (10);
   newformat VARCHAR2 (10);
   myimage SI_StillImage;
   currentformat VARCHAR2 (10);
BEGIN
  SELECT product_photo INTO myimage FROM PM.SI_MEDIA
    WHERE product_id = 1 FOR UPDATE;
  origformat := myimage.SI_Format;
  DBMS_OUTPUT.PUT_LINE('Original format is ' || origformat);
  newformat := 'WBMP';
  myimage.SI_ChangeFormat(newformat);
  INSERT INTO PM.SI_MEDIA (product_id, product_photo) VALUES (44, myimage);
  SELECT product_photo INTO myimage FROM PM.SI_MEDIA
    WHERE product_id = 44;
  currentformat := myimage.SI_Format;
  DBMS_OUTPUT.PUT_LINE('New format is ' || currentformat);
  COMMIT;
END;
/
```

Convert the format of an SI_StillImage object to GIFF using the SI_ConvertFormat( ) procedure:

```
DECLARE
   origformat VARCHAR2 (10);
   newformat VARCHAR2 (10);
```

```
   myimage SI_StillImage;
   currentformat VARCHAR2 (10);
BEGIN
  SELECT product_photo INTO myimage FROM PM.SI_MEDIA
    WHERE product_id = 1 FOR UPDATE;
  origformat := myimage.SI_Format;
  DBMS_OUTPUT.PUT_LINE('Original format is ' || origformat);
  newformat := 'GIFF';
  SI_ConvertFormat(myimage, newformat);
  INSERT INTO PM.SI_MEDIA (product_id, product_photo) VALUES (5, myimage);
  SELECT product_photo INTO myimage FROM PM.SI_MEDIA
    WHERE product_id = 3 FOR UPDATE;
  currentformat := myimage.SI_Format;
  DBMS_OUTPUT.PUT_LINE('New format is ' || currentformat);
  COMMIT;
END;
/
```

SI_Content( )

Format

SI_Content ( )

RETURN BLOB DETERMINISTIC;

Format of Equivalent SQL Function

SI_GetContent(image IN SI_StillImage)

RETURN BLOB DETERMINISTIC;

Description

Returns the BLOB stored in the content_SI attribute of the SI_StillImage object to which this method is applied.

Parameters

None.

Usage Notes

None.

Method Pragmas

PRAGMA RESTRICT_REFERENCES(SI_Content, WNDS, WNPS, RNDS, RNPS)

Function Pragmas

PRAGMA RESTRICT_REFERENCES(SI_GetContent, WNDS, WNPS, RNDS, RNPS)

Exceptions

None.

Examples

Get the BLOB data stored in the content_SI attribute of an SI_StillImage object using the SI_Content( ) method:

```
DECLARE
   myimage SI_StillImage;
```

```
   photo   BLOB;
BEGIN
  SELECT product_photo INTO myimage FROM PM.SI_MEDIA
    WHERE product_id = 1;
  photo := myimage.SI_Content;
END;
/
```

Get the BLOB data stored in the content_SI attribute of an SI_StillImage object using the SI_GetContent( ) function:

```
DECLARE
   myimage SI_StillImage;
   photo   BLOB;
BEGIN
  SELECT product_photo INTO myimage FROM PM.SI_MEDIA
    WHERE product_id = 1;
  photo := SI_GetContent(myimage);
END;
/
```

SI_ContentLength( )

Format

```
SI_ContentLength ( )
 RETURN INTEGER DETERMINISTIC;
```

Format of Equivalent SQL Function

```
SI_GetContentLngth(image IN SI_StillImage)
 RETURN INTEGER DETERMINISTIC;
```

Description

Returns the value (in bytes) of the contentLength_SI attribute of the specified SI_StillImage object.

Parameters

image
The image for which the content length is returned.

Usage Notes

None.

Method Pragmas

PRAGMA RESTRICT_REFERENCES(SI_ContentLength, WNDS, WNPS, RNDS, RNPS)

Function Pragmas

PRAGMA RESTRICT_REFERENCES(SI_GetContentLngth, WNDS, WNPS, RNDS, RNPS)

Exceptions

None.

Examples

Get the value of the contentLength_SI attribute of an SI_StillImage object using the SI_ContentLength( ) method:

```
DECLARE
   length number;
   myimage SI_StillImage;
BEGIN
  SELECT product_photo INTO myimage FROM PM.SI_MEDIA WHERE product_id = 1;
  length := myimage.SI_ContentLength;
  DBMS_OUTPUT.PUT_LINE('Length is ' || length);
END;
/
```

Get the value of the contentLength_SI attribute of an SI_StillImage object using the SI_GetContentLngth( ) function:

```
DECLARE
  length number;
   myimage SI_StillImage;
BEGIN
  SELECT product_photo INTO myimage FROM PM.SI_MEDIA WHERE product_id = 1;
  length := SI_GetContentLngth(myimage);
  DBMS_OUTPUT.PUT_LINE('length is ' || length);
END;
/
```

SI_Format( )

Format

SI_Format ( )

RETURN VARCHAR2 DETERMINISTIC;

Format of Equivalent SQL Function

SI_GetFormat(image IN SI_StillImage)

RETURN VARCHAR2 DETERMINISTIC;

Description

Returns the value of the format_SI attribute (such as TIFF or JFIF) of the SI_StillImage object to which this method is applied.

Parameters

None.

Usage Notes

None.

Method Pragmas

PRAGMA RESTRICT_REFERENCES(SI_Format, WNDS, WNPS, RNDS, RNPS)

Function Pragmas

PRAGMA RESTRICT_REFERENCES(SI_GetFormat, WNDS, WNPS, RNDS, RNPS)

Exceptions

None.

Examples

Get the value of the format_SI attribute of an SI_StillImage object using the SI_Format( ) method:

```
DECLARE
   format VARCHAR2 (10);
```

```
   myimage SI_StillImage;
BEGIN
  SELECT product_photo INTO myimage FROM PM.SI_MEDIA WHERE product_id = 1;
  format := myimage.SI_Format;
  DBMS_OUTPUT.PUT_LINE('Format is ' || format);
END;
/
```

Get the value of the format_SI attribute of an SI_StillImage object using the SI_GetFormat( ) function:

```
DECLARE
   format VARCHAR2 (10);
   myimage SI_StillImage;
BEGIN
  SELECT product_photo INTO myimage FROM PM.SI_MEDIA WHERE product_id = 1;
  format := SI_GetFormat(myimage);
  DBMS_OUTPUT.PUT_LINE('Format is ' || format);
END;
/
```

SI_Height( )

Format

```
SI_Height ( )
 RETURN INTEGER DETERMINISTIC;
```

Format of Equivalent SQL Function

```
SI_GetHeight(image IN SI_StillImage)
 RETURN INTEGER DETERMINISTIC;
```

Description

Returns the value of the height_SI attribute (in pixels) of the SI_StillImage object to which this method is applied.

Parameters

image
The image for which the height is returned.

Usage Notes

None.

Method Pragmas

PRAGMA RESTRICT_REFERENCES(SI_Height, WNDS, WNPS, RNDS, RNPS)

Function Pragmas

PRAGMA RESTRICT_REFERENCES(SI_GetHeight, WNDS, WNPS, RNDS, RNPS)

Exceptions

None.

Examples

Get the value of the height_SI attribute of an SI_StillImage object using the SI_Height( ) method:

```
DECLARE
```

```
   ht NUMBER;
   myimage SI_StillImage;
BEGIN
  SELECT product_photo INTO myimage FROM PM.SI_MEDIA WHERE product_id = 1;
  ht := myimage.SI_Height;
  DBMS_OUTPUT.PUT_LINE('height ' || ht);
END;
/
```

Get the value of the height_SI attribute of an SI_StillImage object using the SI_GetHeight( ) function:

```
DECLARE
  ht NUMBER;
   myimage SI_StillImage;
BEGIN
  SELECT product_photo INTO myimage FROM PM.SI_MEDIA WHERE product_id = 1;
  ht := SI_GetHeight(myimage);
  DBMS_OUTPUT.PUT_LINE('height ' || ht);
END;
/
```

SI_RetainFeatures( )

Format

SI_RetainFeatures( );

RETURN BOOLEAN DETERMINISTIC;

Description

Returns a Boolean value (TRUE or FALSE) to indicate whether or not image features will be extracted and cached.

This method is not in the first edition of the SQL/MM Still Image standard, but has been accepted for inclusion in the next version.

Parameters

None.

Usage Notes

None.

Method Pragma

PRAGMA RESTRICT_REFERENCES(WNDS, WNPS, RNDS, RNPS)

Exceptions

None.

Examples

Determine whether or not the image features will be extracted and cached for a given SI_StillImage object:

```
DECLARE
  myimage     SI_StillImage;
BEGIN
  SELECT product_photo INTO myimage FROM PM.SI_MEDIA
     WHERE product_id = 1;
  IF (myimage.SI_RetainFeatures = TRUE) THEN
  DBMS_OUTPUT.PUT_LINE('Images features will be extracted and cached');
 ELSE
  DBMS_OUTPUT.PUT_LINE('Images features will not be extracted and cached');
```

```
 END IF;
END;
/
```

SI_SetContent( )

Format

SI_SetContent(content IN BLOB);

Format of Equivalent SQL Procedure

SI_ChgContent(image IN OUT NOCOPY SI_StillImage,

content IN BLOB);

Description

Updates the content of an SI_StillImage object. It sets the values of the following attributes:

- content_SI is updated with the value specified with the specified image.
- contentLength_SI is updated with the new content length extracted from the specified image.
- height_SI is updated with the new height extracted from the specified image.
- width_SI is updated with the new width extracted from the specified image.

Parameters

content
The image data. The format of this image data must be the same as the format of the current image.

image
The image whose content you want to update.

Usage Notes

None.

Pragmas

None.

Exceptions

ORDImageSIExceptions.NULL_CONTENT

This exception is raised if the content parameter is NULL.

Examples

Update the content of an SI_StillImage object using the SI_SetContent( ) method:

```
DECLARE
   newcontent BLOB;
   fils BFILE := BFILENAME('FILE_DIR','keyboard.jpg');
   myimage SI_StillImage;
BEGIN
   -- Put the blob in a temporary LOB:
   DBMS_LOB.CREATETEMPORARY(newcontent, TRUE);
   DBMS_LOB.FILEOPEN(fils, DBMS_LOB.FILE_READONLY);
   DBMS_LOB.LOADFROMFILE(newcontent, fils, DBMS_LOB.GETLENGTH(fils));
   DBMS_LOB.FILECLOSE(fils);
   -- Select a row to be updated:
  SELECT product_photo INTO myimage FROM PM.SI_MEDIA WHERE product_id = 1
     FOR UPDATE;
  myimage.SI_SetContent(newcontent);
  -- Update the content of product_photo:
  UPDATE PM.SI_MEDIA
   SET product_photo = myimage where product_id=1;
 -- Free the temporary LOB:
 DBMS_LOB.FREETEMPORARY(newcontent);
 COMMIT;
END;
/
```

Update the content of an SI_StillImage object using the SI_ChgContent( ) procedure:

```
DECLARE
   newcontent BLOB;
   fils BFILE := BFILENAME('FILE_DIR','keyboard.jpg');
   myimage SI_StillImage;

BEGIN
   -- Put the blob in a temporary LOB:
   DBMS_LOB.CREATETEMPORARY(newcontent, TRUE);
   DBMS_LOB.FILEOPEN(fils, DBMS_LOB.FILE_READONLY);
   DBMS_LOB.LOADFROMFILE(newcontent, fils, DBMS_LOB.GETLENGTH(fils));
   DBMS_LOB.FILECLOSE(fils);
   -- Select a row to be updated:
   SELECT product_photo INTO myimage FROM PM.SI_MEDIA WHERE product_id = 1
     FOR UPDATE;
   SI_ChgContent(myimage, newcontent);
```

```
   -- Update the content of product_photo:
   UPDATE PM.SI_MEDIA
   SET product_photo = myimage where product_id=1;
   -- Free the temporary LOB:
   DBMS_LOB.FREETEMPORARY(newcontent);
   COMMIT;
END;
/
```

SI_Thumbnail( )

Format

```
SI_Thumbnail ( )
 RETURN SI_StillImage;
```

Format of Equivalent SQL Function

```
SI_GetThmbnl (image IN SI_StillImage)
 RETURN SI_StillImage;
```

Description

Derives a thumbnail image from the specified SI_StillImage object. The default thumbnail size is 80 by 80 pixels. Because this method preserves the image aspect ratio, the resulting thumbnail size will be as close to 80 by 80 pixels as possible.

Parameters

image
The image for which you want to generate a thumbnail image.

Usage Notes

None.

Pragmas

None.

Exceptions

None.

Examples

Create a new thumbnail image from an SI_StillImage object using the SI_Thumbnail( ) method:

```
DECLARE
   myimage SI_StillImage;
   myThumbnail SI_StillImage;
   height number;
```

```
   width number;
BEGIN
  SELECT product_photo INTO myimage FROM PM.SI_MEDIA WHERE product_id = 1;
  width := myimage.SI_width;
  height := myimage.SI_height;
  DBMS_OUTPUT.PUT_LINE('Height is ' || height || ' pixels.');
  DBMS_OUTPUT.PUT_LINE('Width is ' || width ||  ' pixels.');
  myThumbnail := myimage.SI_Thumbnail;
  width := myThumbnail.SI_width;
  height := myThumbnail.SI_height;
  DBMS_OUTPUT.PUT_LINE('Height is ' || height || ' pixels.');
  DBMS_OUTPUT.PUT_LINE('Width is ' || width ||  ' pixels.');
END;
/
```

Create a new thumbnail image from an SI_StillImage object using the SI_GetThmbnl( ) function:

```
DECLARE
   myimage SI_StillImage;
   myThumbnail SI_StillImage;
   height number;
   width number;
BEGIN
  SELECT product_photo INTO myimage FROM PM.SI_MEDIA WHERE product_id = 1;
  width := myimage.SI_width;
  height := myimage.SI_height;
  DBMS_OUTPUT.PUT_LINE('Height is ' || height || ' pixels.');
  DBMS_OUTPUT.PUT_LINE('Width is ' || width ||  ' pixels.');
  myThumbnail := SI_GetThmbnl(myimage);
  width := myThumbnail.SI_width;
  height := myThumbnail.SI_height;
  DBMS_OUTPUT.PUT_LINE('Height is ' || height || ' pixels.');
  DBMS_OUTPUT.PUT_LINE('Width is ' || width ||  ' pixels.');
END;
/
```

SI_Thumbnail(height,width)

Format

```
SI_Thumbnail(height IN INTEGER, width IN INTEGER)
 RETURN SI_StillImage DETERMINISTIC;
```

Format of Equivalent SQL Function

```
SI_GetSizedThmbnl(image IN SI_StillImage,
                  height  IN INTEGER,
                  width   IN INTEGER) RETURN SI_StillImage DETERMINISTIC;
```

Description

Derives a new thumbnail image from the specified SI_StillImage object using the height and width that you specify. This method does not preserve the aspect ratio.

Parameters

height
The height that you want *inter*Media to use for the thumbnail image.

image
The image for which you want to generate a thumbnail image.

width
The width that you want *inter*Media to use for the thumbnail image.

Usage Notes

To preserve the aspect ratio, supply the appropriate height and width values. To obtain the appropriate height and width values, multiply the image height and width values by the required scaling factor. For example, if an image size is 100 by 100 pixels and the resulting thumbnail image needs to be one fourth of the original image, then the height argument should be 100 divided by 2 and the width argument should be 100 divided by 2. The resulting thumbnail image would be 50 by 50 pixels, and the aspect ratio would be preserved.

Pragmas

None.

Exceptions

None.

Examples

Create a new thumbnail image from an SI_StillImage object with the specified height and width using the SI_Thumbnail(height, width) method:

```
DECLARE
   myimage SI_StillImage;
   myThumbnail SI_StillImage;
   height number;
   width number;
BEGIN
  SELECT product_photo INTO myimage FROM PM.SI_MEDIA WHERE product_id = 1;
  width := myimage.SI_width;
  height := myimage.SI_height;
  DBMS_OUTPUT.PUT_LINE('Height is ' || height || ' pixels.');
  DBMS_OUTPUT.PUT_LINE('Width is ' || width ||  ' pixels.');
  myThumbnail := myimage.SI_Thumbnail(129,121);
  width := myThumbnail.SI_width;
  height := myThumbnail.SI_height;
  DBMS_OUTPUT.PUT_LINE('Height is ' || height || ' pixels.');
  DBMS_OUTPUT.PUT_LINE('Width is ' || width ||  ' pixels.');
END;
/
```

Create a thumbnail image from an SI_StillImage object with the specified height and width using the SI_GetSizedThmbnl function:

```
DECLARE
   myimage SI_StillImage;
   myThumbnail SI_StillImage;
   height number;
   width number;
BEGIN
  SELECT product_photo INTO myimage FROM PM.SI_MEDIA WHERE product_id = 1;
  width := myimage.SI_width;
  height := myimage.SI_height;
  DBMS_OUTPUT.PUT_LINE('Height is ' || height || ' pixels.');
  DBMS_OUTPUT.PUT_LINE('Width is ' || width ||  ' pixels.');
  myThumbnail := SI_GetSizedThmbnl(myimage,129,121);
  width := myThumbnail.SI_width;
  height := myThumbnail.SI_height;
  DBMS_OUTPUT.PUT_LINE('Height is ' || height || ' pixels.');
```

```
  DBMS_OUTPUT.PUT_LINE('Width is ' || width ||  ' pixels.');
END;
/
```

SI_Width( )

Format

SI_Width ( )

RETURN INTEGER DETERMINISTIC;

Format of Equivalent SQL Function

SI_GetWidth(image IN SI_StillImage)

RETURN INTEGER DETERMINISTIC;

Description

Returns the value of the width_SI attribute (in pixels) of the SI_StillImage object to which this method is applied.

Parameters

image
The image for which the width is returned.

Usage Notes

None.

Method Pragmas

PRAGMA RESTRICT_REFERENCES(SI_Width, WNDS, WNPS, RNDS, RNPS)

Function Pragmas

PRAGMA RESTRICT_REFERENCES(SI_GetWidth, WNDS, WNPS, RNDS, RNPS)

Exceptions

None.

Examples

Get the value of the width_SI attribute of an SI_StillImage object using the SI_Width( ) method:

```
DECLARE
```

```
   width NUMBER;
   myimage SI_StillImage;
BEGIN
  SELECT product_photo INTO myimage FROM PM.SI_MEDIA WHERE product_id = 1;
  width := myimage.SI_Width;
  DBMS_OUTPUT.PUT_LINE('Width is ' || width);
END;
/
```

Get the value of the width_SI attribute of an SI_StillImage object using the SI_GetWidth( ) function:

```
DECLARE
   width NUMBER;
   myimage SI_StillImage;
BEGIN
  SELECT product_photo INTO myimage FROM PM.SI_MEDIA WHERE product_id = 1;
  width := SI_GetWidth(myimage);
  DBMS_OUTPUT.PUT_LINE('Width is ' || width);
END;
/
```

SI_Texture Object Type

Describes the image texture characteristics by the size of repeating items (coarseness), brightness variations (contrast), and predominant direction (directionality). This object type is created in the ORDSYS schema with invoker rights. It is declared as an INSTANTIABLE and NOT FINAL type. (See "Internal Helper Types" on page 7-133 for the textureEncoding attribute syntax.)

> **Note:** Use the SI_Texture constructor and method rather than accessing attributes directly to protect yourself from changes to the internal representation of the SI_Texture object.

The SI_Texture object is described as follows:

```
CREATE OR REPLACE TYPE SI_Texture
  AUTHID CURRENT_USER
  AS OBJECT
  (
     --attributes
     SI_TextureEncoding  textureEncoding,
     --Methods
     CONSTRUCTOR FUNCTION SI_Texture
     (sourceImage IN SI_StillImage)
     RETURN SELF AS RESULT DETERMINISTIC,
     --
     MEMBER FUNCTION SI_Score
      (SELF  IN SI_Texture,
       image IN SI_StillImage)
     RETURN DOUBLE PRECISION DETERMINISTIC
) INSTANTIABLE
  NOT FINAL;
/
```

where:

- SI_TextureEncoding: a varray that represents the image texture characteristics such as coarseness, contrast, and directionality.

SI_Texture Constructor

This section describes the SI_Texture object constructor, which is as follows:

- SI_Texture( ) on page 7-125

SI_Texture( )

Format

SI_Texture(sourceImage IN SI_StillImage)

RETURN SELF AS RESULT DETERMINISTIC;

Format of Equivalent SQL Function

SI_FindTexture(sourceImage IN SI_StillImage)

RETURN SI_Texture DETERMINISTIC;

Description

Constructs an SI_Texture object from the specified image.

Parameters

sourceImage
The image whose texture feature is being extracted.

Pragmas

None.

Exceptions

None.

Usage Notes

An error is returned if any of the following conditions is true:

- The value of specified image is NULL.
- The value of sourceImage.SI_Content is NULL.
- The texture feature is not supported for the format of the specified image. This is determined by looking up the SI_IMAGE_FORMAT_FEATURES view or SI_IMAGE_FRMT_FTRS view.

Examples

Create an SI_Texture object using the SI_Texture( ) constructor:

```
DECLARE
   myTexture  SI_Texture;
   myimage    SI_StillImage;
BEGIN
  SELECT product_photo INTO myimage FROM pm.si_media
   WHERE product_id=1 FOR UPDATE;
  myTexture := NEW SI_Texture(myimage);
  UPDATE pm.si_media SET texture = myTexture WHERE product_id=1;
  COMMIT;
END;
/
```

Create an SI_Texture object using the SI_FindTexture( ) function:

```
DECLARE
   myTexture  SI_Texture;
   myimage    SI_StillImage;
BEGIN
   SELECT product_photo INTO myimage FROM pm.si_media
     WHERE product_id=2 FOR UPDATE;
   myTexture := SI_FindTexture(myimage);
   UPDATE pm.si_media SET texture = myTexture WHERE product_id=2;
   COMMIT;
END;
/
```

SI_Texture Method

This section presents reference information on the SI_Texture method used for image matching, which is as follows:

SI_Score( ) for SI_Texture

Format

```
SI_Score(image IN SI_StillImage)
    RETURN DOUBLE PRECISION DETERMINISTIC;
```

Format of Equivalent SQL Function

```
SI_ScoreByTexture(feature IN SI_Texture,
                  image IN SI_StillImage),
RETURN DOUBLE PRECISION DETERMINISTIC;
```

Description

Determines and returns the score of the specified image as compared to the SI_Texture object to which you are applying the method. The lower the returned value, the better the texture of the image is characterized by the SI_Texture value used for scoring the image. This method returns a DOUBLE PRECISION value between 0 and 100, unless any one of the following is true, in which case a NULL value is returned:

- The value of the SI_Texture object to which you apply this method is NULL.
- The value of the specified image is NULL.
- The value of image.SI_Contents is NULL.
- The texture feature is not supported for the specified image.

Parameters

feature
The feature value to be compared with the texture of the specified image.

image
The image whose texture feature is extracted and used for score comparison.

Usage Notes

None.

Pragmas

None.

Exceptions

None.

Examples

Compare an image to an SI_Texture object using the SI_Score( ) method and return the score:

```
DECLARE
    score DOUBLE PRECISION;
    myimage SI_StillImage;
    myTexture SI_Texture;
BEGIN
   SELECT texture INTO myTexture FROM pm.si_media
      WHERE product_id=1;
   SELECT product_photo INTO myimage FROM pm.si_media
      WHERE product_id=2;
   score := myTexture.SI_Score(myimage);
   DBMS_OUTPUT.PUT_LINE('Score is ' || score);
END;
/
```

Compare the texture of an image to an SI_Texture object using the SI_ScoreByTexture( ) function and return the score:

```
DECLARE
    score DOUBLE PRECISION;
    myimage SI_StillImage;
    myTexture SI_Texture;
BEGIN
   SELECT texture INTO myTexture FROM pm.si_media
      WHERE product_id=1;
   SELECT product_photo INTO myimage FROM pm.si_media
      WHERE product_id=2;
   score := SI_ScoreByTexture(myTexture, myimage);
   DBMS_OUTPUT.PUT_LINE('Score is ' || score);
END;
/
```

Views

The schema, SI_INFORMTN_SCHEMA, contains several views that identify the supported image formats and implementation-defined values. The privilege set on these views is PUBLIC WITH GRANT OPTION. The views are:

- SI_IMAGE_FORMATS
- SI_IMAGE_FORMAT_CONVERSIONS
- SI_IMAGE_FORMAT_FEATURES
- SI_THUMBNAIL_FORMATS
- SI_VALUES

The column names, data types, and a description is provided for each of these views in the tables that follow.

Table 7–1 describes the SI_IMAGE_FORMATS view. This view identifies the supported image formats.

Table 7–1 SI_IMAGE_FORMATS View

Column Name	Data Type	Description
SI_FORMAT	VARCHAR2(SI_MaxFormatLength)	A list of the supported image formats.

Table 7–2 describes the SI_IMAGE_FORMAT_CONVERSIONS view. This view identifies the source and target image formats for which an image format conversion is supported. The short name for this view is SI_IMAGE_FORMAT_CONVRSNS.

Table 7–2 SI_IMAGE_FORMAT_CONVERSIONS View

Column Name	Data Type	Description
SI_SOURCE_FORMAT	VARCHAR2(SI_MaxFormatLength)	The format of the source image.
SI_TARGET_FORMAT	VARCHAR2(SI_MaxFormatLength)	The format of the target image.

Table 7–3 describes the SI_IMAGE_FORMAT_FEATURES view. This view identifies the image formats for which a basic feature is supported. The short name for this view is SI_IMAGE_FRMT_FTRS.

Table 7–3 SI_IMAGE_FORMAT_FEATURES View

Column Name	Data Type	Description
SI_FORMAT	VARCHAR2(SI_MaxFormatLength)	The format name.
SI_FEATURE_NAME	VARCHAR2(100)	The basic feature name that is supported by the named format. Value can be any of the following: ■ SI_AverageColor ■ SI_Texture ■ SI_PositionalColor ■ SI_ColorHistogram

Table 7–4 describes the SI_THUMBNAIL_FORMATS view. This view identifies the image formats from which thumbnail images can be derived. The short name for this view is SI_THUMBNAIL_FRMTS.

Table 7–4 SI_THUMBNAIL_FORMATS View

Column Name	Data Type	Description
SI_FORMAT	VARCHAR2(SI_MaxFormatLength)	The formats from which a thumbnail image can be derived.

Table 7–5 describes the SI_VALUES view. This view identifies the implementation-defined values.

Table 7–5 SI_VALUES View

Column Name	Data Type	Description
SI_VALUE	VARCHAR2(SI_MaxFormatLength)	The implementation-defined meta-variables. The SI_VALUES view has 8 rows where each row has one of the following SI_VALUE column values: ■ SI_MaxContentLength is the maximum length for the binary representation of the SI_StillImage attribute. ■ SI_MaxFeatureNameLength is the maximum length for the character representation of a basic image feature name. ■ SI_MaxFormatLength is the maximum length for the character representation of an image format indication. ■ SI_MaxHistogramLength is the maximum number of color/frequency pairs that are admissible in an SI_ColorHistogram feature value. ■ SI_MaxRGBColor is the maximum value for each component of a color value that is represented by the RGB color space. ■ SI_MaxTextureLength is the number of bytes needed for the encoded representation of an SI_Texture object. ■ SI_MaxValueLength is the maximum length for the character representation (name) of the meta-variables in the SI_VALUES view. ■ SI_NumberSections is the number of most significant color values that are represented by the SI_PositionalColor value.
SI_SUPPORTED_ VALUE	NUMBER(38)	A column with the following values: ■ 0 If the implementation places no limit on the meta-variable defined by SI_VALUE column or cannot determine the limit. ■ NULL If the implementation does not support any features for which the meta-variable is applicable. ■ Any non-NULL, nonzero value The maximum size supported by the implementation for this meta-variable.

Internal Helper Types

An attribute that consists of an array is specified as an internal helper type. Internal helper types are created in the ORDSYS schema and do not have public synonyms.

The internal helper types are the following:

- colorsList

  The syntax for this internal helper type is:

  ```
  CREATE OR REPLACE TYPE colorsList AS VARRAY(100) OF SI_Color;
  ```

  This internal helper type is used to specify the SI_ColorsList attribute of the SI_ColorHistogram Object Type as described on page 7-20.

- colorFrequenciesList

  The syntax for this internal helper type is:

  ```
  CREATE OR REPLACE TYPE colorFrequenciesList AS VARRAY(100) OF DOUBLE
  PRECISION;
  ```

  This internal helper type is used to specify the SI_FrequenciesList attribute of the SI_ColorHistogram Object Type as described on page 7-20.

- colorPositions

  The syntax for this internal helper type is:

  ```
  CREATE OR REPLACE TYPE colorPositions AS VARRAY(9) OF SI_Color;
  ```

  This internal helper type is used to specify the SI_ColorPositions attribute of the SI_PositionalColor Object Type as described on page 7-72.

- textureEncoding

  The syntax for this internal helper type is:

  ```
  CREATE OR REPLACE TYPE textureEncoding AS VARRAY(5) of DOUBLE PRECISION;
  ```

  This internal helper type is used to specify the SI_TextureEncoding attribute of the SI_Texture Object Type as described on page 7-123.

8

ORDVideo

Oracle *inter*Media *("inter*Media") contains the following information about the ORDVideo object type:

The examples in this chapter use the ONLINE_MEDIA table in the Product Media sample schema. To replicate the examples on your own computer, you should begin with the examples shown in the reference pages for the ORDVideo constructors and the import( ) and importFrom( ) methods. Substitute video files you have for the ones shown in the examples. In addition, for a user "ron" to use the examples, the following statements must be issued before ron executes the examples, where "/mydir/work" is the directory where ron will find the video data:

```
CONNECT /as sysdba
CREATE OR REPLACE DIRECTORY FILE_DIR as '/mydir/work';
GRANT READ ON DIRECTORY FILE_DIR TO PUBLIC WITH GRANT OPTION;
```

See *Oracle Database Sample Schemas* for information on the sample schemas.

> **Note:** If you manipulate the video data itself (by either directly modifying the BLOB or changing the external source), then you must ensure that the object attributes stay synchronized and the update time is modified; otherwise, the object attributes will not match the video data.

Methods invoked at the ORDSource level that are handed off to the source plug-in for processing have ctx (RAW) as the first argument. Before calling any of these

methods for the first time, the client should allocate the ctx structure, initialize it to NULL, and invoke the openSource( ) method. At this point, the source plug-in can initialize context for this client. When processing is complete, the client should invoke the closeSource( ) method.

Methods invoked from a source plug-in call have the first argument as ctx (RAW).

Methods invoked at the ORDVideo level that are handed off to the format plug-in for processing have ctx (RAW) as the first argument. Before calling any of these methods for the first time, the client should allocate the ctx structure and initialize it to NULL.

Note: In the current release, none of the plug-ins provided by Oracle and not all source or format plug-ins will use the ctx argument, but if you code as previously described, your application should work with current or future source or format plug-ins.

You should use any of the individual set methods to set the attribute value for an object for formats not natively supported; otherwise, for formats natively supported, use the setProperties( ) method to populate the attributes of the object or write a format plug-in.

ORDVideo Object Type

The ORDVideo object type supports the storage and management of video data. This object type is defined as follows:

```
CREATE OR REPLACE TYPE ORDVideo
AS OBJECT
(
  -- ATTRIBUTES
description         VARCHAR2(4000),
source              ORDSource,
format              VARCHAR2(31),
mimeType            VARCHAR2(4000),
comments            CLOB,
  -- VIDEO RELATED ATTRIBUTES
width               INTEGER,
height              INTEGER,
frameResolution     INTEGER,
frameRate           INTEGER,
videoDuration       INTEGER,
numberOfFrames      INTEGER,
compressionType     VARCHAR2(4000),
numberOfColors      INTEGER,
bitRate             INTEGER,

  -- METHODS
-- CONSTRUCTORS
--
STATIC FUNCTION init( ) RETURN ORDVideo,
STATIC FUNCTION init(srcType     IN VARCHAR2,
                     srcLocation IN VARCHAR2,
                     srcName     IN VARCHAR2) RETURN ORDVideo,
-- Methods associated with the date attribute
MEMBER FUNCTION getUpdateTime RETURN DATE,
PRAGMA RESTRICT_REFERENCES(getUpdateTime, WNDS, WNPS, RNDS, RNPS),
MEMBER PROCEDURE setUpdateTime(current_time DATE),
-- Methods associated with the description attribute
MEMBER PROCEDURE setDescription(user_description IN VARCHAR2),
MEMBER FUNCTION getDescription RETURN VARCHAR2,
PRAGMA RESTRICT_REFERENCES(getDescription, WNDS, WNPS, RNDS, RNPS),

-- Methods associated with the mimeType attribute
MEMBER PROCEDURE setMimeType(mime IN VARCHAR2),
```

```
MEMBER FUNCTION getMimeType RETURN VARCHAR2,
PRAGMA RESTRICT_REFERENCES(getMimeType, WNDS, WNPS, RNDS, RNPS),

-- Methods associated with the source attribute
MEMBER FUNCTION processSourceCommand(
                                  ctx       IN OUT RAW,
                                  cmd       IN VARCHAR2,
                                  arguments IN VARCHAR2,
                                  result    OUT RAW)
               RETURN RAW,

MEMBER FUNCTION  isLocal RETURN BOOLEAN,
PRAGMA RESTRICT_REFERENCES(isLocal, WNDS, WNPS, RNDS, RNPS),

MEMBER PROCEDURE  setLocal,
MEMBER PROCEDURE  clearLocal,

MEMBER PROCEDURE setSource(
                          source_type     IN VARCHAR2,
                          source_location IN VARCHAR2,
                          source_name     IN VARCHAR2),
MEMBER FUNCTION getSource RETURN VARCHAR2,
PRAGMA RESTRICT_REFERENCES(getSource, WNDS, WNPS, RNDS, RNPS),

MEMBER FUNCTION getSourceType RETURN VARCHAR2,
PRAGMA RESTRICT_REFERENCES(getSourceType, WNDS, WNPS, RNDS, RNPS),

MEMBER FUNCTION getSourceLocation RETURN VARCHAR2,
PRAGMA RESTRICT_REFERENCES(getSourceLocation, WNDS, WNPS, RNDS, RNPS),

MEMBER FUNCTION getSourceName RETURN VARCHAR2,
PRAGMA RESTRICT_REFERENCES(getSourceName, WNDS, WNPS, RNDS, RNPS),

MEMBER PROCEDURE import(ctx IN OUT RAW),
MEMBER PROCEDURE importFrom(
                           ctx             IN OUT RAW,
                           source_type     IN VARCHAR2,
                           source_location IN VARCHAR2,
                           source_name     IN VARCHAR2),
MEMBER PROCEDURE export(
                       ctx             IN OUT RAW,
                       source_type     IN VARCHAR2,
                       source_location IN VARCHAR2,
                       source_name     IN VARCHAR2),
```

```
MEMBER FUNCTION getContentLength(ctx IN OUT RAW) RETURN INTEGER,
PRAGMA RESTRICT_REFERENCES(getContentLength, WNDS, WNPS, RNDS, RNPS),

MEMBER PROCEDURE getContentInLob(
                              ctx      IN OUT RAW,
                              dest_lob IN OUT NOCOPY BLOB,
                              mimeType OUT VARCHAR2,
                              format   OUT VARCHAR2),

MEMBER FUNCTION getContent RETURN BLOB,
PRAGMA RESTRICT_REFERENCES(getContent, WNDS, WNPS, RNDS, RNPS),

MEMBER PROCEDURE deleteContent,

MEMBER FUNCTION getBFILE RETURN BFILE,
PRAGMA RESTRICT_REFERENCES(getBFILE, WNDS, WNPS, RNDS, RNPS),

-- Methods associated with file operations on the source
MEMBER FUNCTION openSource(userArg IN RAW, ctx OUT RAW) RETURN INTEGER,
MEMBER FUNCTION closeSource(ctx IN OUT RAW) RETURN INTEGER,
MEMBER FUNCTION trimSource(ctx     IN OUT RAW,
                           newlen  IN INTEGER) RETURN INTEGER,
MEMBER PROCEDURE readFromSource(
                              ctx      IN OUT RAW,
                              startPos IN INTEGER,
                              numBytes IN OUT INTEGER,
                              buffer   OUT RAW),
MEMBER PROCEDURE writeToSource(
                             ctx      IN OUT RAW,
                             startPos IN INTEGER,
                             numBytes IN OUT INTEGER,
                             buffer   IN RAW),

-- Methods associated with the video attributes accessors
MEMBER PROCEDURE setFormat(knownformat IN VARCHAR2),
MEMBER FUNCTION getFormat RETURN VARCHAR2,
PRAGMA RESTRICT_REFERENCES(getFormat, WNDS, WNPS, RNDS, RNPS),

MEMBER PROCEDURE setFrameSize(knownWidth IN INTEGER, knownHeight IN INTEGER),
MEMBER PROCEDURE getFrameSize(retWidth OUT INTEGER, retHeight OUT INTEGER),
PRAGMA RESTRICT_REFERENCES(getFrameSize, WNDS, WNPS, RNDS, RNPS),

MEMBER PROCEDURE setFrameResolution(knownFrameResolution IN INTEGER),
MEMBER FUNCTION getFrameResolution RETURN INTEGER,
PRAGMA RESTRICT_REFERENCES(getFrameResolution, WNDS, WNPS, RNDS, RNPS),
```

```
MEMBER PROCEDURE setFrameRate(knownFrameRate IN INTEGER),
MEMBER FUNCTION getFrameRate RETURN INTEGER,
PRAGMA RESTRICT_REFERENCES(getFrameRate, WNDS, WNPS, RNDS, RNPS),

MEMBER PROCEDURE setVideoDuration(knownVideoDuration IN INTEGER),
MEMBER FUNCTION getVideoDuration RETURN INTEGER,
PRAGMA RESTRICT_REFERENCES(getVideoDuration, WNDS, WNPS, RNDS, RNPS),

MEMBER PROCEDURE setNumberOfFrames(knownNumberOfFrames IN INTEGER),
MEMBER FUNCTION getNumberOfFrames RETURN INTEGER,
PRAGMA RESTRICT_REFERENCES(getNumberOfFrames, WNDS, WNPS, RNDS, RNPS),

MEMBER PROCEDURE setCompressionType(knownCompressionType IN VARCHAR2),
MEMBER FUNCTION getCompressionType RETURN VARCHAR2,
PRAGMA RESTRICT_REFERENCES(getCompressionType, WNDS, WNPS, RNDS, RNPS),

MEMBER PROCEDURE setNumberOfColors(knownNumberOfColors IN INTEGER),
MEMBER FUNCTION getNumberOfColors RETURN INTEGER,
PRAGMA RESTRICT_REFERENCES(getNumberOfColors, WNDS, WNPS, RNDS, RNPS),

MEMBER PROCEDURE setBitRate(knownBitRate IN INTEGER),
MEMBER FUNCTION getBitRate RETURN INTEGER,
PRAGMA RESTRICT_REFERENCES(getBitRate, WNDS, WNPS, RNDS, RNPS),

MEMBER PROCEDURE setKnownAttributes(
                                 knownFormat IN VARCHAR2,
                                 knownWidth IN INTEGER,
                                 knownHeight IN INTEGER,
                                 knownFrameResolution IN INTEGER,
                                 knownFrameRate IN INTEGER,
                                 knownVideoDuration IN INTEGER
                                 knownNumberOfFrames IN INTEGER,
                                 knownCompressionType IN VARCHAR2,
                                 knownNumberOfColors IN INTEGER,
                                 knownBitRate IN INTEGER),

-- Methods associated with setting all the properties
MEMBER PROCEDURE setProperties(ctx         IN OUT RAW,
                               setComments IN BOOLEAN),
MEMBER FUNCTION checkProperties(ctx IN OUT RAW) RETURN BOOLEAN,

MEMBER FUNCTION getAttribute(
                           ctx  IN OUT RAW,
                           name IN VARCHAR2) RETURN VARCHAR2,
```

```
MEMBER PROCEDURE getAllAttributes(
                                  ctx        IN OUT RAW,
                                  attributes IN OUT NOCOPY CLOB),

-- Methods associated with video processing
MEMBER FUNCTION processVideoCommand(
                                    ctx       IN OUT RAW,
                                    cmd       IN VARCHAR2,
                                    arguments IN VARCHAR2,
                                    result    OUT RAW)
                    RETURN RAW
);
```

where:

- description: the description of the video object.
- source: the ORDSource where the video data is to be found.
- format: the format in which the video data is stored.
- mimeType: the MIME type information.
- comments: the metadata information of the video object.
- width: the width of each frame of the video data.
- height: the height of each frame of the video data.
- frameResolution: the frame resolution of the video data.
- frameRate: the frame rate of the video data.
- videoDuration: the total duration of the video data stored.
- numberOfFrames: the number of frames in the video data.
- compressionType: the compression type of the video data.
- numberOfColors: the number of colors in the video data.
- bitRate: the bit rate of the video data.

> **Note:** The comments attribute is populated by the setProperties( ) method when the setComments parameter is TRUE and by Oracle *inter*Media Annotator. Oracle Corporation recommends that you not write to this attribute directly.

ORDVideo Constructors

This section describes the *inter*Media constructor functions, which are as follow:

init( ) for ORDVideo

Format

init( ) RETURN ORDVideo;

Description

Initializes instances of the ORDVideo object type.

Parameters

None.

Pragmas

None.

Exceptions

None.

Usage Notes

This constructor is a static method that initializes all the ORDVideo attributes to NULL with the following exceptions:

- source.updateTime is set to SYSDATE
- source.local is set to 1 (local)
- source.localData is set to empty_blob

You should begin using the init( ) method as soon as possible to allow you to more easily initialize the ORDVideo object type, especially if the ORDVideo type evolves and attributes are added in a future release. INSERT statements left unchanged using the default constructor (which initializes each object attribute), will fail under these circumstances.

Examples

Initialize the ORDVideo object attributes:

```
BEGIN
 INSERT INTO pm.online_media (product_id, product_video)
 VALUES (2004, ORDSYS.ORDVideo.init());
```

```
 COMMIT;
END;
/
```

init(srcType,srcLocation,srcName) for ORDVideo

Format

```
init(srcType     IN VARCHAR2,
   srcLocation IN VARCHAR2,
   srcName     IN VARCHAR2)
 RETURN ORDVideo;
```

Description

Initializes instances of the ORDVideo object type.

Parameters

srcType
The source type of the video data.

srcLocation
The source location of the video data.

srcName
The source name of the video data.

Pragmas

None.

Exceptions

None.

Usage Notes

This constructor is a static method that initializes all the ORDVideo attributes to NULL with the following exceptions:

- source.updateTime is set to SYSDATE
- source.local is set to 0
- source.localData is set to empty_blob

- source.srcType is set to the input value
- source.srcLocation is set to the input value
- source.srcName is set to the input value

You should begin using the init( ) method as soon as possible to allow you to more easily initialize the ORDVideo object type, especially if the ORDVideo type evolves and attributes are added in a future release. INSERT statements left unchanged using the default constructor (which initializes each object attribute), will fail under these circumstances.

Examples

Initialize the ORDVideo object attributes:

```
BEGIN
 INSERT INTO pm.online_media (product_id, product_video)
 VALUES (2030, ORDSYS.ORDVideo.init('FILE', 'FILE_DIR','speakers.rm'));
 COMMIT;
END;
/
```

ORDVideo Methods

This section presents reference information on the Oracle *inter*Media methods used specifically for video data manipulation.

Chapter 3 presents reference information on the Oracle *inter*Media methods that are common to ORDAudio, ORDDoc, ORDImage, and ORDVideo. Use the methods presented in both chapters to get and set attributes, and to perform metadata extractions.

For more information on object types and methods, see *Oracle Database Concepts*.

The following methods are presented in this section:

checkProperties( )

Format

checkProperties(ctx IN OUT RAW) RETURN BOOLEAN;

Description

Checks all the properties of the stored video data, including the following video attributes: format, width, height, frame resolution, frame rate, video duration, number of frames, compression type, number of colors, and bit rate.

Parameters

ctx
The format plug-in context information.

Usage Notes

The checkProperties( ) method does not check the MIME type because a file can have multiple correct MIME types and this is not well defined.

Pragmas

None.

Exceptions

ORDVideoExceptions.VIDEO_PLUGIN_EXCEPTION

This exception is raised if you call the checkProperties( ) method and the video plug-in raises an exception when calling this method.

Examples

Check property information for known video attributes:

```
DECLARE
 obj ORDSYS.ORDVideo;
 ctx RAW(64) :=NULL;
BEGIN
 SELECT p.product_video INTO obj FROM pm.online_media p
   WHERE p.product_id  = 2030;
 IF(obj.checkProperties(ctx)) THEN
```

```
  DBMS_OUTPUT.PUT_LINE('check Properties returned true');
 ELSE
  DBMS_OUTPUT.PUT_LINE('check Properties returned false');
 END IF;
 COMMIT;
 EXCEPTION
  WHEN ORDSYS.ORDVideoExceptions.VIDEO_PLUGIN_EXCEPTION THEN
   DBMS_OUTPUT.PUT_LINE('ORDVideoExceptions.VIDEO_PLUGIN_EXCEPTION caught');
  WHEN OTHERS THEN
   DBMS_OUTPUT.PUT_LINE('EXCEPTION caught');
END;
/
```

getAllAttributes( )

Format

```
getAllAttributes(
                ctx          IN OUT RAW,
                attributes IN OUT NOCOPY CLOB);
```

Description

Returns a formatted string for convenient client access. For natively supported formats, the string includes the following list of audio data attributes separated by a comma (,): width, height, format, frameResolution, frameRate, videoDuration, numberOfFrames, compressionType, numberOfColors, and bitRate. For user-defined formats, the string is defined by the format plug-in.

Parameters

ctx
The format plug-in context information.

attributes
The attributes.

Usage Notes

Generally, these video data attributes are available from the header of the formatted video data.

Video data attribute information can be extracted from the video data itself. You can extend support to a video format that is not understood by the ORDVideo object by implementing an ORDPLUGINS.ORDX_<format>_VIDEO package that supports that format. See *Oracle interMedia User's Guide* for more information.

Pragmas

None.

Exceptions

ORDVideoExceptions.METHOD_NOT_SUPPORTED

This exception is raised if you call the getAllAttributes( ) method and the video plug-in raises an exception when calling this method.

Examples

Return all video attributes for video data stored in the database:

```
DECLARE
 obj ORDSYS.ORDVideo;
 tempLob   CLOB;
 ctx RAW(64) :=NULL;
BEGIN
 SELECT p.product_video INTO obj FROM pm.online_media p
  WHERE p.product_id  = 2030;
 DBMS_OUTPUT.PUT_LINE('getting comma separated list of all attributes');
 DBMS_OUTPUT.PUT_LINE('-------------------------------------------');
 DBMS_LOB.CREATETEMPORARY(tempLob, FALSE, DBMS_LOB.CALL);
 obj.getAllAttributes(ctx,tempLob);
 DBMS_OUTPUT.PUT_LINE(DBMS_LOB.substr(tempLob, DBMS_LOB.getLength(tempLob),1));
 COMMIT;
 EXCEPTION
  WHEN ORDSYS.ORDVideoExceptions.METHOD_NOT_SUPPORTED THEN
   DBMS_OUTPUT.PUT_LINE('VIDEO METHOD_NOT_SUPPORTED EXCEPTION caught');
  WHEN OTHERS THEN
   DBMS_OUTPUT.PUT_LINE('EXCEPTION CAUGHT');
END;
/
```

getAttribute( )

Format

```
getAttribute(
        ctx     IN OUT RAW,
        name IN VARCHAR2)
RETURN VARCHAR2;
```

Description

Returns the value of the requested attribute from video data for user-defined formats only.

> **Note:** This method is supported only for user-defined format plug-ins.

Parameters

ctx
The format plug-in context information.

name
The name of the attribute.

Usage Notes

None.

Pragmas

None.

Exceptions

ORDVideoExceptions.VIDEO_PLUGIN_EXCEPTION

This exception is raised if you call the getAttribute( ) method and the video plug-in raises an exception when calling this method.

Examples

Return information for the specified video attribute for video data stored in the database. (Because this example uses a supported data format, rather than a user-written plug-in, an exception will be raised.)

```
DECLARE
 obj ORDSYS.ORDVideo;
 res VARCHAR2(4000);
 ctx RAW(64) :=NULL;
BEGIN
 SELECT p.product_video INTO obj FROM pm.online_media p
  WHERE p.product_id  = 2030;
 DBMS_OUTPUT.PUT_LINE('getting video duration');
 DBMS_OUTPUT.PUT_LINE('---------------------');
 res := obj.getAttribute(ctx,'video_duration');
 COMMIT;
 EXCEPTION
  WHEN ORDSYS.ORDVideoExceptions.VIDEO_PLUGIN_EXCEPTION THEN
   DBMS_OUTPUT.PUT_LINE('VIDEO PLUGIN EXCEPTION caught');
  WHEN OTHERS THEN
   DBMS_OUTPUT.PUT_LINE('EXCEPTION caught');
END;
/
```

getBitRate( )

Format

getBitRate( ) RETURN INTEGER;

Description

Returns the value of the bitRate attribute of the video object.

Parameters

None.

Usage Notes

None.

Pragmas

PRAGMA RESTRICT_REFERENCES(getBitRate, WNDS, WNPS, RNDS, RNPS)

Exceptions

None.

Examples

Return the object attribute value of the bitRate attribute of the video object:

```
DECLARE
 obj ORDSYS.ORDVideo;
 res INTEGER;
BEGIN
 SELECT p.product_video INTO obj FROM pm.online_media p
  WHERE p.product_id  = 2030;
 res := obj.getBitRate();
 DBMS_OUTPUT.PUT_LINE('bit rate : ' || res );
 COMMIT;
END;
/
```

getCompressionType( )

Format

getCompressionType( ) RETURN VARCHAR2;

Description

Returns the value of the compressionType attribute of the video object.

Parameters

None.

Usage Notes

None.

Pragmas

PRAGMA RESTRICT_REFERENCES(getCompressionType, WNDS, WNPS, RNDS, RNPS)

Exceptions

None.

Examples

Return the object attribute value of the compressionType attribute of the video object:

```
DECLARE
 obj ORDSYS.ORDVideo;
 res VARCHAR2(4000);
BEGIN
 SELECT p.product_video INTO obj FROM pm.online_media p
  WHERE p.product_id  = 2030;
 res := obj.getCompressionType();
 DBMS_OUTPUT.PUT_LINE('compression type: ' ||res);
 COMMIT;
END;
/
```

getContentInLob( )

Format

```
getContentInLob(
                ctx             IN OUT RAW,
                dest_lob        IN OUT NOCOPY BLOB,
                mimeType  OUT VARCHAR2,
                format          OUT VARCHAR2);
```

Description

Copies data from a data source into the specified BLOB. The BLOB must not be the BLOB in the source.localData attribute (of the embedded ORDSource object).

Parameters

ctx
The source plug-in context information.

dest_lob
The LOB in which to receive data.

mimeType
The MIME type of the data; this may or may not be returned.

format
The format of the data; this may or may not be returned.

Usage Notes

None.

Pragmas

None.

Exceptions

ORDSourceExceptions.METHOD_NOT_SUPPORTED

This exception is raised if you call the getContentInLob( ) method and this method is not supported by the source plug-in being used.

ORDSourceExceptions.SOURCE_PLUGIN_EXCEPTION

This exception is raised if you call the getContentInLob( ) method and the source plug-in raises an exception.

See Appendix F for more information about these exceptions.

Examples

Get data from a data source into the specified BLOB on the local source:

```
DECLARE
 obj ORDSYS.ORDVideo;
 tempBLob BLOB;
 mimeType VARCHAR2(4000);
 format VARCHAR2(31);
 ctx RAW(64) :=NULL;
BEGIN
 SELECT p.product_video INTO obj FROM pm.online_media p
  WHERE p.product_id  = 2030;
 IF (obj.isLocal) THEN
  DBMS_OUTPUT.PUT_LINE('local is true');
 END IF;
 DBMS_LOB.CREATETEMPORARY(tempBLob, true, 10);
 obj.getContentInLob(ctx,tempBLob, mimeType,format);
 DBMS_OUTPUT.PUT_LINE('Length is ' ||TO_CHAR(DBMS_LOB.getLength(tempBLob)));
 COMMIT;
 EXCEPTION
  WHEN ORDSYS.ORDSourceExceptions.METHOD_NOT_SUPPORTED THEN
   DBMS_OUTPUT.PUT_LINE('ORDSourceExceptions.METHOD_NOT_SUPPORTED caught');
  WHEN OTHERS THEN
   DBMS_OUTPUT.PUT_LINE('EXCEPTION caught');
END;
/
```

getContentLength( )

Format

```
getContentLength(ctx IN OUT RAW) RETURN INTEGER;
```

Description

Returns the length of the video data content stored in the source.

Parameters

ctx
The source plug-in context information.

Usage Notes

None.

Pragmas

PRAGMA RESTRICT_REFERENCES(getContentLength, WNDS, WNPS, RNDS, RNPS)

Exceptions

ORDSourceExceptions.INCOMPLETE_SOURCE_INFORMATION

This exception is raised if you call the getContentLength( ) method and the value of source.srcType attribute is NULL.

ORDSourceExceptions.SOURCE_PLUGIN_EXCEPTION

This exception is raised if you call the getContentLength( ) method and the source plug-in raises an exception.

See Appendix F for more information about this exception.

Examples

See the example in import( ) on page 8-36.

getDescription( )

Format

getDescription( ) RETURN VARCHAR2;

Description

Returns the description of the video data.

Parameters

None.

Usage Notes

None.

Pragmas

PRAGMA RESTRICT_REFERENCES(getDescription, WNDS, WNPS, RNDS, RNPS)

Exceptions

ORDVideoExceptions.DESCRIPTION_IS_NOT_SET

This exception is raised if you call the getDescription( ) method and the description attribute is not set.

Examples

See the example in setDescription( ) on page 8-44.

getFormat( )

Format

getFormat( ) RETURN VARCHAR2;

Description

Returns the value of the format attribute of the video object.

Parameters

None.

Usage Notes

None.

Pragmas

PRAGMA RESTRICT_REFERENCES(getFormat, WNDS, WNPS, RNDS, RNPS)

Exceptions

ORDVideoExceptions.VIDEO_FORMAT_IS_NULL

This exception is raised if you call the getFormat( ) method and the value for the format attribute is NULL.

Examples

Get the format for some stored video data:

```
DECLARE
 obj ORDSYS.ORDVideo;
BEGIN
 SELECT p.product_video INTO obj FROM pm.online_media p
  WHERE p.product_id  = 2030;
 DBMS_OUTPUT.PUT_LINE('writing format');
 DBMS_OUTPUT.PUT_LINE('--------------');
 DBMS_OUTPUT.PUT_LINE(obj.getFormat);
 COMMIT;
END;
/
```

getFrameRate( )

Format

getFrameRate( ) RETURN INTEGER;

Description

Returns the value of the frameRate attribute of the video object.

Parameters

None.

Usage Notes

None.

Pragmas

PRAGMA RESTRICT_REFERENCES(getFrameRate, WNDS, WNPS, RNDS, RNPS)

Exceptions

None.

Examples

Return the object attribute value of the frame rate for video data stored in the database:

```
DECLARE
 obj ORDSYS.ORDVideo;
 res INTEGER;
BEGIN
 SELECT p.product_video INTO obj FROM pm.online_media p
  WHERE p.product_id  = 2030;
 res := obj.getFrameRate();
 DBMS_OUTPUT.PUT_LINE('frame rate : ' ||res);
 COMMIT;
END;
/
```

getFrameResolution( )

Format

getFrameResolution( ) RETURN INTEGER;

Description

Returns the value of the frameResolution attribute of the video object.

Parameters

None.

Usage Notes

None.

Pragmas

PRAGMA RESTRICT_REFERENCES(getFrameResolution, WNDS, WNPS, RNDS, RNPS)

Exceptions

None.

Examples

Return the value of the frame resolution for the video data:

```
DECLARE
 obj ORDSYS.ORDVideo;
 res INTEGER;
BEGIN
 SELECT p.product_video INTO obj FROM pm.online_media p
  WHERE p.product_id  = 2030;
 res := obj.getFrameResolution();
 DBMS_OUTPUT.PUT_LINE('resolution : ' ||res);
 COMMIT;
END;
/
```

getFrameSize( )

Format

```
getFrameSize(retWidth OUT INTEGER,
             retHeight OUT INTEGER);
```

Description

Returns the value of the height and width attributes of the video object.

Parameters

retWidth
The frame width in pixels.

retHeight
The frame height in pixels.

Usage Notes

None.

Pragmas

PRAGMA RESTRICT_REFERENCES(getFrameSize, WNDS, WNPS, RNDS, RNPS)

Exceptions

None.

Examples

Return the frame size (width and height) for video data:

```
DECLARE
 obj ORDSYS.ORDVideo;
 width INTEGER;
 height INTEGER;
BEGIN
 SELECT p.product_video INTO obj FROM pm.online_media p
  WHERE p.product_id  = 2030;
 obj.getFrameSize(width, height);
 DBMS_OUTPUT.PUT_LINE('width :' || width);
```

```
 DBMS_OUTPUT.PUT_LINE('height :' || height);
 COMMIT;
END;
/
```

getNumberOfColors( )

Format

getNumberOfColors( ) RETURN INTEGER;

Description

Returns the value of the numberOfColors attribute of the video object.

Parameters

None.

Usage Notes

None.

Pragmas

PRAGMA RESTRICT_REFERENCES(getNumberOfColors, WNDS, WNPS, RNDS, RNPS)

Exceptions

None.

Examples

Return the object attribute value of the numberOfColors attribute of the video object:

```
DECLARE
 obj ORDSYS.ORDVideo;
 res INTEGER;
BEGIN
 SELECT p.product_video INTO obj FROM pm.online_media p
  WHERE p.product_id  = 2030;
 res := obj.getNumberOfColors();
 DBMS_OUTPUT.PUT_LINE('number of colors: ' ||res);
 COMMIT;
END;
/
```

getNumberOfFrames( )

Format

getNumberOfFrames( ) RETURN INTEGER;

Description

Returns the value of the numberOfFrames attribute of the video object.

Parameters

None.

Usage Notes

None.

Pragmas

PRAGMA RESTRICT_REFERENCES(getNumberOfFrames, WNDS, WNPS, RNDS, RNPS)

Exceptions

None.

Examples

Return the object attribute value of the total number of frames in the video data:

```
DECLARE
 obj ORDSYS.ORDVideo;
 res INTEGER;
BEGIN
 SELECT p.product_video INTO obj FROM pm.online_media p
  WHERE p.product_id  = 2030;
 res := obj.getNumberOfFrames();
 DBMS_OUTPUT.PUT_LINE('number of frames : ' ||res);
 COMMIT;
END;
/
```

getVideoDuration( )

Format

getVideoDuration( ) RETURN INTEGER;

Description

Returns the value of the videoDuration attribute of the video object.

Parameters

None.

Usage Notes

None.

Pragmas

PRAGMA RESTRICT_REFERENCES(getVideoDuration, WNDS, WNPS, RNDS, RNPS)

Exceptions

None.

Examples

Return the total time to play the video data:

```
DECLARE
 obj ORDSYS.ORDVideo;
 res INTEGER;
BEGIN
 SELECT p.product_video INTO obj FROM pm.online_media p
  WHERE p.product_id  = 2030;
 res := obj.getVideoDuration();
 DBMS_OUTPUT.PUT_LINE('video duration : ' ||res);
 COMMIT;
END;
/
```

import( )

Format

```
import(ctx IN OUT RAW);
```

Description

Transfers video data from an external video data source to a local source (localData) within Oracle Database.

Parameters

ctx
The source plug-in context information. This should be allocated and initialized to NULL. If you are using a user-defined source plug-in, you should call the openSource( ) method. See the introduction to this chapter for more information.

Usage Notes

Use the setSource( ) method to set the source.srcType, source.srcLocation, and source.srcName attributes (of the embedded ORDSource object) for the external video data source prior to calling the import( ) method.

After importing data from an external video data source to a local source (within Oracle Database), the source information remains unchanged (that is, pointing to the source from where the data was imported).

Invoking this method implicitly calls the setUpdateTime( ) and setLocal( ) methods.

This method is invoked at the ORDSource level, which uses the PL/SQL UTL_HTTP package to import media data from an HTTP data source. You can use environment variables to specify the proxy behavior of the UTL_HTTP package. For example, on UNIX, setting the environment variable http_proxy to a URL specifies that the UTL_HTTP package will use that URL as the proxy server for HTTP requests. Setting the no_proxy environment variable to a domain name specifies that the HTTP proxy server will not be used for URLs in the specified domain.

See *PL/SQL Packages and Types Reference* for more information about the UTL_HTTP PL/SQL package.

Pragmas

None.

Exceptions

ORDSourceExceptions.INCOMPLETE_SOURCE_INFORMATION

This exception is raised if you call the import( ) method and the value of the source.srcType attribute is NULL.

ORDSourceExceptions.NULL_SOURCE

This exception is raised if you call the import( ) method and the value of the source.localData attribute is NULL.

ORDSourceExceptions.METHOD_NOT_SUPPORTED

This exception is raised if you call the import( ) method and this method is not supported by the source plug-in being used.

ORDSourceExceptions.SOURCE_PLUGIN_EXCEPTION

This exception is raised if you call the import( ) method and the source plug-in raises an exception.

See Appendix F for more information about these exceptions.

Examples

Import video data by first setting the source and then importing it:

```
DECLARE
 obj ORDSYS.ORDVideo;
 ctx RAW(64) :=NULL;
BEGIN
 SELECT p.product_video INTO obj FROM pm.online_media p
  WHERE p.product_id = 2030 FOR UPDATE;
 DBMS_OUTPUT.PUT_LINE('setting and getting source');
 DBMS_OUTPUT.PUT_LINE('--------------------------');
 -- Set source to a file:
 obj.setSource('file','FILE_DIR','speakers.rm');
 -- Get source information:
 DBMS_OUTPUT.PUT_LINE(obj.getSource());
 -- Import data:
 obj.import(ctx);
 -- Check size:
 DBMS_OUTPUT.PUT_LINE('Length is ' || TO_CHAR(obj.getContentLength(ctx)));
 UPDATE pm.online_media p SET p.product_video = obj WHERE p.product_id = 2030;
 COMMIT;
END;
/
```

importFrom( )

Format

```
importFrom(ctx IN OUT RAW,
           source_type      IN VARCHAR2,
           source_location  IN VARCHAR2,
           source_name      IN VARCHAR2);
```

Description

Transfers video data from the specified external video data source to a local source (localData) within Oracle Database.

Parameters

ctx
The source plug-in context information. This should be allocated and initialized to NULL. If you are using a user-defined source plug-in, you should call the openSource( ) method. See the introduction to this chapter for more information.

source_type
The type of the source video data.

source_location
The location from which the source video data is to be imported.

source_name
The name of the source video data.

Usage Notes

This method is similar to the import( ) method except the source information is specified as parameters to the method instead of separately.

You must ensure that the directory indicated with the source_location parameter exists or is created before you use this method for srcType "file".

After importing data from an external video data source to a local source (within Oracle Database), the source information (that is, pointing to the source from where the data was imported) is set to the input values.

Invoking this method implicitly calls the setUpdateTime( ) and setLocal( ) methods.

Pragmas

None.

Exceptions

ORDSourceExceptions.NULL_SOURCE exception

This exception is raised if you call the importFrom( ) method and the value the source.localData attribute is NULL.

ORDSourceExceptions.METHOD_NOT_SUPPORTED

This exception is raised if you call the importFrom( ) method and this method is not supported by the source plug-in being used.

ORDSourceExceptions.SOURCE_PLUGIN_EXCEPTION

This exception is raised if you call the importFrom( ) method and the source plug-in raises an exception.

See Appendix F for more information about these exceptions.

Examples

Import video data from the specified external data source into the local source:

```
DECLARE
  obj ORDSYS.ORDVideo;
  ctx RAW(64) :=NULL;
BEGIN
  SELECT p.product_video INTO obj FROM pm.online_media p
     WHERE p.product_id = 2004 FOR UPDATE;
  DBMS_OUTPUT.PUT_LINE('setting and getting source');
  DBMS_OUTPUT.PUT_LINE('--------------------------');
  -- Import data:
  obj.importFrom(ctx,'file','FILE_DIR','speakers.rm');
  -- Check size:
  DBMS_OUTPUT.PUT_LINE('Length is ' ||TO_CHAR(obj.getContentLength(ctx)));
  UPDATE pm.online_media p SET p.product_video = obj WHERE p.product_id = 2004;
  COMMIT;
  EXCEPTION
        WHEN ORDSYS.ORDSourceExceptions.METHOD_NOT_SUPPORTED THEN
         DBMS_OUTPUT.put_line('Source METHOD_NOT_SUPPORTED caught');
        WHEN ORDSYS.ORDSourceExceptions.SOURCE_PLUGIN_EXCEPTION THEN
         DBMS_OUTPUT.put_line('SOURCE PLUGIN EXCEPTION caught');
```

```
        WHEN ORDSYS.ORDVideoExceptions.METHOD_NOT_SUPPORTED THEN
         DBMS_OUTPUT.put_line('VIDEO METHOD_NOT_SUPPORTED EXCEPTION caught');
        WHEN ORDSYS.ORDVideoExceptions.VIDEO_PLUGIN_EXCEPTION THEN
         DBMS_OUTPUT.put_line('VIDEO PLUGIN EXCEPTION caught');
        WHEN OTHERS THEN
         DBMS_OUTPUT.PUT_LINE('EXCEPTION Caught');
END;
/
```

processVideoCommand( )

Format

```
processVideoCommand(ctx          IN OUT RAW,
                    cmd          IN VARCHAR2,
                    arguments IN VARCHAR2,
                    result       OUT RAW)
RETURN RAW;
```

Description

Lets you send a command and related arguments to the format plug-in for processing.

> **Note:** This method is supported only for user-defined format plug-ins.

Parameters

ctx
The format plug-in context information.

cmd
Any command recognized by the format plug-in.

arguments
The arguments of the command.

result
The result of calling this method returned by the format plug-in.

Usage Notes

Use this method to send any video commands and their respective arguments to the format plug-in. Commands are not interpreted; they are taken and passed through to a format plug-in to be processed.

If the format is set to NULL, then the processVideoCommand( ) method uses the default format plug-in; otherwise, it uses your user-defined format plug-in.

You can extend support to a format that is not understood by the ORDVideo object by preparing an ORDPLUGINS.ORDX_<format>_VIDEO package that supports that format. See *Oracle interMedia User's Guide* for more information.

Pragmas

None.

Exceptions

ORDVideoExceptions.METHOD_NOT_SUPPORTED

ORDVideoExceptions.VIDEO_PLUGIN_EXCEPTION

Either exception is raised if you call the processVideoCommand( ) method and the video plug-in raises an exception when calling this method.

Examples

None.

setBitRate( )

Format

```
setBitRate(knownBitRate IN INTEGER);
```

Description

Sets the value of the bitRate attribute of the video object.

Parameters

knownBitRate
The bit rate.

Usage Notes

Calling this method implicitly calls the setUpdateTime( ) method.

Pragmas

None.

Exceptions

ORDVideoExceptions.NULL_INPUT_VALUE

This exception is raised if you call the setBitRate( ) method and the value for the knownBitRate parameter is NULL.

Examples

See the example in setFrameSize( ) on page 8-50.

setCompressionType( )

Format

setCompressionType(knownCompressionType IN VARCHAR2);

Description

Sets the value of the compressionType attribute of the video object.

Parameters

knownCompressionType
A known compression type.

Usage Notes

Calling this method implicitly calls the setUpdateTime( ) method.

Pragmas

None.

Exceptions

ORDVideoExceptions.NULL_INPUT_VALUE

This exception is raised if you call the setCompressionType( ) method and the value for the knownCompressionType parameter is NULL.

Examples

See the example in setFrameSize( ) on page 8-50.

setDescription( )

Format

setDescription (user_description IN VARCHAR2);

Description

Sets the description of the video data.

Parameters

user_description
The description of the video data.

Usage Notes

Each video object may need a description to help some client applications. For example, a Web-based client can show a list of video descriptions from which a user can select one to access the video data.

Web access components and other client components provided with Oracle *inter*Media make use of this description attribute to present video data to users.

Calling this method implicitly calls the setUpdateTime( ) method.

Pragmas

None.

Exceptions

None.

Examples

Set the description attribute for some video data:

```
DECLARE
 obj ORDSYS.ORDVideo;
BEGIN
 SELECT p.product_video INTO obj FROM pm.online_media p
  WHERE p.product_id = 2030 FOR UPDATE;
 DBMS_OUTPUT.PUT_LINE('writing description');
 DBMS_OUTPUT.PUT_LINE('-------------');
```

```
 obj.setDescription('This is a video of a speaker');
 DBMS_OUTPUT.PUT_LINE(obj.getDescription());
 UPDATE pm.online_media p SET p.product_video = obj WHERE p.product_id = 2688;
 COMMIT;
END;
/
```

setFormat( )

Format

setFormat(knownFormat IN VARCHAR2);

Description

Sets the format attribute of the video object.

Parameters

knownFormat
The known format of the video data to be set in the video object.

Usage Notes

Calling this method implicitly calls the setUpdateTime( ) method.

Pragmas

None.

Exceptions

ORDVideoExceptions.NULL_INPUT_VALUE

This exception is raised if you call the setFormat( ) method and the value for the knownFormat parameter is NULL.

Examples

Set the format for some stored video data:

```
DECLARE
 obj ORDSYS.ORDVideo;
BEGIN
 SELECT p.product_video INTO obj FROM pm.online_media p
  WHERE p.product_id = 2030 FOR UPDATE;
 DBMS_OUTPUT.PUT_LINE('current format');
 DBMS_OUTPUT.PUT_LINE('--------------');
 DBMS_OUTPUT.PUT_LINE(obj.getFormat);
 obj.setFormat('rm');
 DBMS_OUTPUT.PUT_LINE('new format');
 DBMS_OUTPUT.PUT_LINE('--------------');
```

```
DBMS_OUTPUT.PUT_LINE(obj.getFormat);
 UPDATE pm.online_media p SET p.product_video = obj
   WHERE p.product_id = 2030;
COMMIT;
EXCEPTION
 WHEN ORDSYS.ORDVideoExceptions.NULL_INPUT_VALUE THEN
  DBMS_OUTPUT.PUT_LINE('ORDVideoExceptions.NULL_INPUT_VALUE caught');
 WHEN OTHERS THEN
  DBMS_OUTPUT.PUT_LINE('EXCEPTION caught');
END;
/
```

setFrameRate( )

Format

setFrameRate(knownFrameRate IN INTEGER);

Description

Sets the value of the frameRate attribute of the video object.

Parameters

knownFrameRate
The frame rate.

Usage Notes

Calling this method implicitly calls the setUpdateTime( ) method.

Pragmas

None.

Exceptions

ORDVideoExceptions.NULL_INPUT_VALUE

This exception is raised if you call the setFrameRate( ) method and the value for the knownFrameRate parameter is NULL.

Examples

See the example in setFrameSize( ) on page 8-50.

setFrameResolution( )

Format

setFrameResolution(knownFrameResolution IN INTEGER);

Description

Sets the value of the frameResolution attribute of the video object.

Parameters

knownFrameResolution
The known frame resolution in pixels per inch.

Usage Notes

Calling this method implicitly calls the setUpdateTime( ) method.

Pragmas

None.

Exceptions

ORDVideoExceptions.NULL_INPUT_VALUE

This exception is raised if you call the setFrameResolution( ) method and the value for the knownFrameResolution parameter is NULL.

Examples

See the example in setFrameSize( ) on page 8-50.

setFrameSize( )

Format

```
setFrameSize(knownWidth  IN INTEGER,
             knownHeight IN INTEGER);
```

Description

Sets the value of the height and width attributes of the video object.

Parameters

knownWidth
The frame width in pixels.

knownHeight
The frame height in pixels.

Usage Notes

Calling this method implicitly calls the setUpdateTime( ) method.

Pragmas

None.

Exceptions

ORDVideoExceptions.NULL_INPUT_VALUE

This exception is raised if you call the setFrameSize( ) method and the value for either the knownWidth or knownHeight parameter is NULL.

Examples

Set the frame size (width and height) for video data:

```
DECLARE
 obj ORDSYS.ORDVideo;
BEGIN
 SELECT p.product_video INTO obj FROM pm.online_media p
   WHERE p.product_id = 2030 FOR UPDATE;
 obj.setFrameSize(1,2);
```

```
 obj.setFrameResolution(4);
 obj.setFrameRate(5);
 obj.setVideoDuration(20);
 obj.setNumberOfFrames(8);
 obj.setCompressionType('Cinepak');
 obj.setBitRate(1500);
 obj.setNumberOfColors(256);
 UPDATE pm.online_media p SET p.product_video = obj WHERE p.product_id = 2030;
 COMMIT;
END;
/
```

setKnownAttributes( )

Format

```
setKnownAttributes(
                    knownFormat              IN VARCHAR2,
                    knownWidth               IN INTEGER,
                    knownHeight              IN INTEGER,
                    knownFrameResolution     IN INTEGER,
                    knownFrameRate           IN INTEGER,
                    knownVideoDuration       IN INTEGER,
                    knownNumberOfFrames      IN INTEGER,
                    knownCompressionType     IN VARCHAR2,
                    knownNumberOfColors      IN INTEGER,
                    knownBitRate             IN INTEGER);
```

Description

Sets the known video attributes for the video data.

Parameters

knownFormat
The known format.

knownWidth
The known width.

knownHeight
The known height.

knownFrameResolution
The known frame resolution.

knownFrameRate
The known frame rate.

knownVideoDuration
The known video duration.

knownNumberOfFrames
The known number of frames.

knownCompressionType
The known compression type.

knownNumberOfColors
The known number of colors.

knownBitRate
The known bit rate.

Usage Notes

Calling this method implicitly calls the setUpdateTime( ) method.

Pragmas

None.

Exceptions

None.

Examples

Set the property information for all known attributes for video data:

```
DECLARE
 obj ORDSYS.ORDVideo;
 width integer;
 height integer;
BEGIN
 SELECT p.product_video INTO obj FROM pm.online_media p
  WHERE p.product_id = 2030 FOR UPDATE;
 obj.setKnownAttributes('MOOV',1,2,4,5,20,8,'Cinepak', 256, 1500);
 obj.getFrameSize(width, height);
 DBMS_OUTPUT.PUT_LINE('width: '  || TO_CHAR(width));
 DBMS_OUTPUT.PUT_LINE('height: ' || TO_CHAR(height));
 DBMS_OUTPUT.PUT_LINE('format: ' || obj.getFormat());
 DBMS_OUTPUT.PUT_LINE('frame resolution: ' || TO_CHAR(obj.getFrameResolution()));
 DBMS_OUTPUT.PUT_LINE('frame rate: ' || TO_CHAR(obj.getFrameRate()));
 DBMS_OUTPUT.PUT_LINE('video duration: ' || TO_CHAR(obj.getVideoDuration()));
 DBMS_OUTPUT.PUT_LINE('number of frames: ' || TO_CHAR(obj.getNumberOfFrames()));
```

```
 DBMS_OUTPUT.PUT_LINE('compression type: ' || obj.getCompressionType());
 DBMS_OUTPUT.PUT_LINE('bite rate: ' || TO_CHAR(obj.getBitRate()));
 DBMS_OUTPUT.PUT_LINE('number of colors: ' || TO_CHAR(obj.getNumberOfColors()));
 UPDATE pm.online_media p SET p.product_video = obj
   WHERE p.product_id = 2030;
 COMMIT;
END;
/
```

setNumberOfColors( )

Format

setNumberOfColors(knownNumberOfColors IN INTEGER);

Description

Sets the value of the numberOfColors attribute of the video object.

Parameters

knownNumberOfColors
A known number of colors.

Usage Notes

Calling this method implicitly calls the setUpdateTime( ) method.

Pragmas

None.

Exceptions

ORDVideoExceptions.NULL_INPUT_VALUE

This exception is raised if you call the setNumberOfColors( ) method and the value for the knownNumberOfColors parameter is NULL.

Examples

See the example in setFrameSize( ) on page 8-50.

setNumberOfFrames( )

Format

```
setNumberOfFrames(knownNumberOfFrames IN INTEGER);
```

Description

Sets the value of the numberOfFrames attribute of the video object.

Parameters

knownNumberOfFrames
A known number of frames.

Usage Notes

Calling this method implicitly calls the setUpdateTime( ) method.

Pragmas

None.

Exceptions

ORDVideoExceptions.NULL_INPUT_VALUE

This exception is raised if you call the setNumberOfFrames( ) method and the value for the knownNumberOfFrames parameter is NULL.

Examples

See the example in setFrameSize( ) on page 8-50.

setProperties( )

Format

```
setProperties(ctx IN OUT RAW,
              setComments IN BOOLEAN);
```

Description

Reads the video data to get the values of the object attributes and then stores them in the object. This method sets the properties for each of the following attributes of the video data for which values are available: format, height, width, frame resolution, frame rate, video duration, number of frames, compression type, number of colors, and bit rate. This method populates the comments field of the object with a rich set of format and application properties in XML form if the value of the setComments parameter is TRUE.

Parameters

ctx
The format plug-in context information.

setComments
A Boolean value that indicates whether or not the comments field of the object is populated. If the value is TRUE, then the comments field of the object is populated with a rich set of format and application properties of the video object in XML form; otherwise, if the value is FALSE, the comments field of the object remains unpopulated. The default value is FALSE.

Usage Notes

If the property cannot be extracted from the media source, then the respective attribute is set to NULL.

If the format is set to NULL, then the setProperties( ) method uses the default format plug-in; otherwise, it uses your user-defined format plug-in.

Pragmas

None.

Exceptions

ORDVideoExceptions.VIDEO_PLUGIN_EXCEPTION

This exception is raised if you call the setProperties( ) method and the video plug-in raises an exception when calling this method.

Examples

Set the property information for known video attributes:

```
DECLARE
  obj ORDSYS.ORDVideo;
  ctx RAW(64) :=NULL;
BEGIN
  SELECT p.product_video INTO obj FROM pm.online_media p
    WHERE p.product_id  = 2030 FOR UPDATE;
  obj.setProperties(ctx,FALSE);
  UPDATE pm.online_media p SET p.product_video = obj
    WHERE p.product_id = 2030;
  COMMIT;
  EXCEPTION
   WHEN ORDSYS.ORDVideoExceptions.VIDEO_PLUGIN_EXCEPTION THEN
    DBMS_OUTPUT.PUT_LINE('ORDVideoExceptions.VIDEO_PLUGIN_EXCEPTION caught');
   WHEN OTHERS THEN
    DBMS_OUTPUT.PUT_LINE('exception raised');
END;
/
```

setVideoDuration( )

Format

setVideoDuration(knownVideoDuration IN INTEGER);

Description

Sets the value of the videoDuration attribute of the video object.

Parameters

knownVideoDuration
A known video duration.

Usage Notes

Calling this method implicitly calls the setUpdateTime( ) method.

Pragmas

None.

Exceptions

ORDVideoExceptions.NULL_INPUT_VALUE

This exception is raised if you call the setVideoDuration( ) method and the value for the knownVideoDuration parameter is NULL.

Examples

See the example in setFrameSize( ) on page 8-50.

9

*inter*Media Relational Interface Reference

Application developers, who created multimedia applications without using the Oracle *inter*Media (*"inter*Media") object types to store and manage media data in relational tables, and who do not want to migrate their existing multimedia applications to use *inter*Media objects, can use the *inter*Media relational interface for managing their media data. The *inter*Media relational interface consists of a set of methods for:

- Extracting information directly from their media data as either an XML string or as XML and individual attributes
- Processing and copying image data
- Loading media data into Oracle Database
- Exporting media data from Oracle Database into operating system files

The primary benefit of using the *inter*Media relational interface is to let application developers take advantage of *inter*Media functions with only minimal changes to their applications, and all without having to change their schemas to the *inter*Media objects to store their data.

The Oracle *inter*Media relational interface consists of a set of static methods (see Static Methods for the Relational Interface on page 9-3) for the *inter*Media objects: ORDAudio, ORDDoc, ORDImage, and ORDVideo. Because these are static methods, no object is instantiated. Data is passed by method arguments rather than by object attributes.

Methods related to the source of the media have ctx (RAW) as the first argument. Before calling any of these methods for the first time, the client should allocate the ctx structure and initialize it to NULL.

ORDAudio, ORDDoc, and ORDVideo methods related to media parsing have ctx (RAW) as the first argument. Before calling any of these methods for the first time, the client should allocate the ctx structure and initialize it to NULL.

Static Methods for the Relational Interface

This section presents reference information on the static methods for the relational interface. It is divided into subsections that describe those static methods (export( ), import( ), and importFrom( )) that are common to all object types and those static methods that are unique to a particular object type or are implemented differently for the different object types.

Static Methods Common to All Object Types

The examples in this section assume that you have created the test tables as described in Example Audio Table Definition on page 9-16, Example Media Table Definition on page 9-30, Example Image Table Definition on page 9-41, and Example Video Table Definition on page 9-60, respectively for each object type.

This section presents reference information on the *inter*Media common static methods used for the relational interface.

export( )

Format

```
export(ctx              IN OUT RAW,
       local_data       IN BLOB,
       source_type      IN VARCHAR2,
       source_location  IN VARCHAR2,
       source_name      IN VARCHAR2);
```

Description

Copies data from a local source (local_data) within the database to an external data source.

> **Note:** The export( ) method natively supports only sources of source type "file". User-defined sources may also support the export( ) method.

Parameters

ctx
The source plug-in context information.

local_data
The BLOB location that is being exported.

source_type
The type of the source data that is being exported.

source_location
The location to which the source data is to be exported.

source_name
The name of the object to where the source data is to be exported.

Usage Notes

After calling the export( ) method, you can issue a SQL DELETE statement or call the DBMS_LOB.TRIM procedure to delete the content stored locally, if desired.

The export( ) method for a source type of "file" is similar to a file copy operation in that the original data stored in the BLOB is not touched, other than for reading purposes.

The export( ) method writes only to a database directory object that the user has privilege to access. That is, you can access a directory that you have created using the SQL CREATE DIRECTORY statement, or one to which you have been granted READ access. To execute the CREATE DIRECTORY statement, you must have the CREATE ANY DIRECTORY privilege. In addition, you must use the DBMS_JAVA.GRANT_PERMISSION call to specify to which files the user and ORDSYS can write. The user must be granted the write permission so that he or she can write to the file; ORDSYS must be granted the write permission so that it can export the file on behalf of the user. (The installation procedure creates the ORDSYS user by default during installation. See *Oracle interMedia User's Guide* for more information.)

For example, the following SQL*Plus commands grant the user, MEDIAUSER, and ORDSYS the permission to write to the file named filmtrack1.au:

```
CALL DBMS_JAVA.GRANT_PERMISSION(
        'MEDIAUSER',
        'java.io.FilePermission',
        '/audio/movies/filmtrack1.au',
        'write');
CALL DBMS_JAVA.GRANT_PERMISSION(
        'ORDSYS',
        'java.io.FilePermission',
        '/audio/movies/filmtrack1.au',
        'write');
```

Pragmas

None.

Exceptions

ORDSourceExceptions.INCOMPLETE_SOURCE_INFORMATION

This exception is raised if you call the export( ) method and the value of the source_type parameter is NULL.

ORDSourceExceptions.METHOD_NOT_SUPPORTED

This exception is raised if you call the export( ) method and this method is not supported by the source plug-in being used.

ORDSourceExceptions.SOURCE_PLUGIN_EXCEPTION

This exception is raised if you call the export( ) method and the source plug-in raises an exception.

ORDSourceExceptions.IO_ERROR

This exception is raised if the export( ) method encounters an error writing the BLOB data to the file specified.

See Appendix F for more information about these exceptions.

Examples

Export data from a local source to an external audio data source:

> **Note:** *e:/mydir/work/testaud.dat* must be replaced with the file specification of your test file and <*system-password*> with the system password.

```
CONNECT SYSTEM/<system-password>;
CREATE OR REPLACE DIRECTORY AUDIODIR AS 'e:/mydir/work';
GRANT READ ON DIRECTORY AUDIODIR TO PUBLIC WITH GRANT OPTION;

CALL DBMS_JAVA.GRANT_PERMISSION(
    'MEDIAUSER',
    'java.io.FilePermission',
    'e:/mydir/work/testaud.dat',
    'write');
CALL DBMS_JAVA.GRANT_PERMISSION(
  'ORDSYS',
  'java.io.FilePermission',
  '/private1/adetwork/istuart_imedt/work/testaud.dat',
  'write');

CONNECT MEDIAUSER/MEDIAUSER;
DECLARE
  audio_data BLOB;
  ctx RAW(64) :=NULL;
BEGIN
  SELECT aud INTO audio_data FROM taud WHERE N = 1;
  ORDSYS.ORDAudio.export(ctx,audio_data,'file','AUDIODIR','testaud.dat');
```

```
EXCEPTION
  WHEN OTHERS THEN
  RAISE;
END;
/
```

importFrom( )

Format

```
importFrom(ctx              IN OUT RAW,
           local_data       IN OUT NOCOPY BLOB,
           source_type      IN VARCHAR2,
           source_location  IN VARCHAR2,
           source_name      IN VARCHAR2);
```

Description

Transfers data from the specified external data source to the source.localData attribute (of the embedded ORDSource object type) within the database.

Parameters

ctx
The source plug-in context information. This should be allocated and initialized to NULL. If you are using a user-defined source plug-in, you should call the openSource( ) method. See the introduction to this chapter for more information.

local_data
The BLOB location to receive the data.

source_type
The type of the source data.

source_location
The location from which the source data is to be imported.

source_name
The name of the source data.

Usage Notes

You must ensure that the directory indicated by the source_location parameter exists or is created before you use this method for file sources.

Pragmas

None.

Exceptions

ORDSourceExceptions.NULL_SOURCE

This exception is raised if you call the importFrom( ) method and the value of the local_data parameter is NULL or has not been initialized.

ORDSourceExceptions.METHOD_NOT_SUPPORTED

This exception is raised if you call the importFrom( ) method and this method is not supported by the source plug-in being used.

ORDSourceExceptions.SOURCE_PLUGIN_EXCEPTION

This exception is raised if you call the importFrom( ) method and the source plug-in raises an exception.

See Appendix F for more information about these exceptions.

Examples

Import data from the specified external data source into the local source:

> **Note:** *<ORACLE_HOME>* must be replaced with your Oracle home and *<system-password>* with the system password.

```
CONNECT system/<system-password>;
CREATE OR REPLACE DIRECTORY DOCDIR AS 'e:\<ORACLE_HOME>\ord\doc\demo';
GRANT READ ON DIRECTORY DOCDIR TO PUBLIC WITH GRANT OPTION;

CONNECT MEDIAUSER/MEDIAUSER;

DECLARE
  document_data BLOB;
  ctx RAW(64) :=NULL;
BEGIN
   SELECT document INTO document_data FROM tdoc WHERE N = 1 FOR UPDATE;
   ORDSYS.ORDDoc.importFrom(ctx,document_data,'file','DOCDIR','testimg.dat');
   UPDATE tdoc SET document = document_data WHERE N = 1;
   COMMIT;
   SELECT document INTO document_data FROM tdoc WHERE N = 2 FOR UPDATE;
   ORDSYS.ORDDoc.importFrom(ctx,document_data,'file','DOCDIR','testaud.dat');
```

```
   UPDATE tdoc SET document = document_data WHERE N = 2;
   COMMIT;
   SELECT document INTO document_data FROM tdoc WHERE N = 3 FOR UPDATE;
   ORDSYS.ORDDoc.importFrom(ctx,document_data,'file','DOCDIR','testvid.dat');
   UPDATE tdoc SET document = document_data WHERE N = 3;
   COMMIT;
EXCEPTION
   WHEN OTHERS THEN
   RAISE;
END;
/
```

importFrom( ) (all attributes)

Format

```
importFrom(ctx              IN OUT RAW,
           local_data       IN OUT NOCOPY BLOB,
           source_type      IN VARCHAR2,
           source_location  IN VARCHAR2,
           source_name      IN VARCHAR2,
           format           OUT VARCHAR2,
           mime_type        OUT VARCHAR2);
```

Description

Transfers data from the specified external data source to the source.localData attribute (of the embedded ORDSource object type) within the database.

Parameters

ctx
The source plug-in context information.

local_data
The BLOB location to receive the data.

source_type
The type of the source data.

source_location
The location from which the source data is to be imported.

source_name
The name of the source data.

format
The format of the data. The value is returned if it is available (from HTTP sources).

mime_type
The MIME type of the data. The value is returned if it is available (from HTTP sources).

Usage Notes

You must ensure that the directory indicated by the source_location parameter exists or is created before you use this method for file sources.

Pragmas

None.

Exceptions

ORDSourceExceptions.NULL_SOURCE

This exception is raised if you call the importFrom( ) method and the value of the local_data parameter is NULL or has not been initialized.

ORDSourceExceptions.METHOD_NOT_SUPPORTED

This exception is raised if you call the importFrom( ) method and this method is not supported by the source plug-in being used.

ORDSourceExceptions.SOURCE_PLUGIN_EXCEPTION

This exception is raised if you call the importFrom( ) method and the source plug-in raises an exception.

See Appendix F for more information about these exceptions.

Examples

Import image data from the specified external data source into the local source:

> **Note:** *<ORACLE_HOME>* must be replaced with your Oracle home and *<system-password>* with the system password.

```
CONNECT system/<system-password>;
CREATE OR REPLACE DIRECTORY IMAGEDIR AS 'e:\<ORACLE_HOME>\ord\img\demo';
GRANT READ ON DIRECTORY IMAGEDIR TO PUBLIC WITH GRANT OPTION;

DECLARE
  image_data BLOB;
  ctx RAW(64) :=NULL;
```

```
  img_format     VARCHAR2(32)  := NULL;
  img_mime_type  VARCHAR2(80);
BEGIN
   SELECT img INTO image_data FROM timg WHERE N = 1 FOR UPDATE;
   ORDSYS.ORDImage.importFrom(ctx,image_data,'file','IMAGEDIR','testimg.dat',img_format,img_mime_type);
   UPDATE timg SET img = image_data WHERE N = 1;
   COMMIT;
EXCEPTION
   WHEN OTHERS THEN
   RAISE;
END;
/
```

Static Methods Unique to the ORDAudio Object Type Relational Interface

This section presents reference information on the *inter*Media static methods unique to the ORDAudio relational interface.

The relational interface adds *inter*Media support to audio data stored in BLOBs and BFILEs rather than in the ORDAudio object type. The following interface is defined in the ordaspec.sql file:

```
.
.
.
  -- Static Methods for the relational interface
  STATIC PROCEDURE export(ctx             IN OUT RAW,
                          local_data      IN BLOB,
                          source_type     IN VARCHAR2,
                          source_location IN VARCHAR2,
                          source_name     IN VARCHAR2),
  --
  STATIC PROCEDURE importFrom(ctx             IN OUT RAW,
                              local_data      IN OUT NOCOPY BLOB,
                              source_type     IN VARCHAR2,
                              source_location IN VARCHAR2,
                              source_name     IN VARCHAR2),
  --
  STATIC PROCEDURE importFrom(ctx             IN OUT RAW,
                              local_data      IN OUT NOCOPY BLOB,
                              source_type     IN VARCHAR2,
                              source_location IN VARCHAR2,
                              source_name     IN VARCHAR2,
                              format          OUT VARCHAR2,
                              mime_type       OUT VARCHAR2),
  --
  STATIC PROCEDURE getProperties(ctx            IN OUT RAW,
                                 audioBlob      IN BLOB,
                                 attributes     IN OUT NOCOPY CLOB,
                                 format         IN VARCHAR2),
  --
  STATIC PROCEDURE getProperties(ctx              IN OUT RAW,
                                 audioBlob        IN BLOB,
                                 attributes       IN OUT NOCOPY CLOB,
                                 mimeType         OUT VARCHAR2,
                                 format           IN OUT VARCHAR2,
```

```
                                encoding         OUT VARCHAR2,
                                numberOfChannels OUT INTEGER,
                                samplingRate     OUT INTEGER,
                                sampleSize       OUT INTEGER,
                                compressionType  OUT VARCHAR2,
                                audioDuration    OUT INTEGER),
  --
  STATIC PROCEDURE getProperties(ctx            IN OUT RAW,
                                audioBfile      IN OUT NOCOPY BFILE,
                                attributes      IN OUT NOCOPY CLOB,
                                format          IN VARCHAR2),
  --
  STATIC PROCEDURE getProperties(ctx              IN OUT RAW,
                                audioBfile        IN OUT NOCOPY BFILE,
                                attributes        IN OUT NOCOPY CLOB,
                                mimeType          OUT VARCHAR2,
                                format            IN OUT VARCHAR2,
                                encoding          OUT VARCHAR2,
                                numberOfChannels  OUT INTEGER,
                                samplingRate      OUT INTEGER,
                                sampleSize        OUT INTEGER,
                                compressionType   OUT VARCHAR2,
                                audioDuration     OUT INTEGER),
.
.
.
```

Example Audio Table Definition

The methods described in this section show examples based on a test audio table TAUD. Refer to the TAUD table definition that follows when reading through the examples:

TAUD Table Definition

```
CREATE TABLE taud(n                NUMBER,
                  aud              BLOB,
                  attributes       CLOB,
                  mimetype         VARCHAR2(4000),
                  format           VARCHAR2(31),
                  encoding         VARCHAR2(256),
                  numberofchannels INTEGER,
                  samplingrate     INTEGER,
                  samplesize       INTEGER,
                  compressiontype  VARCHAR2(4000),
```

```
                  audioduration    INTEGER)
STORAGE (INITIAL 100K NEXT 100K PCTINCREASE 0);

INSERT INTO taud VALUES(1,EMPTY_BLOB(),EMPTY_CLOB(), NULL, NULL, NULL, NULL,
   NULL, NULL, NULL, NULL);
INSERT INTO taud VALUES(2,EMPTY_BLOB(),EMPTY_CLOB(), NULL, NULL, NULL, NULL,
   NULL, NULL, NULL, NULL);
COMMIT;
```

getProperties( ) for BLOBs

Format

```
getProperties(ctx          IN OUT RAW,
              audioBlob    IN BLOB,
              attributes   IN OUT NOCOPY CLOB,
              format       IN VARCHAR2);
```

Description

Reads the audio BLOB data to get the values of the media attributes for supported formats, and then stores them in the input CLOB. This method populates the CLOB with an extensive set of format and application properties in XML form.

Parameters

ctx
The format plug-in context information.

audioBlob
The audio data represented as a BLOB.

attributes
The CLOB to hold the XML attribute information generated by the getProperties( ) method. This CLOB is populated with an extensive set of format and application properties of the audio BLOB data in XML form.

format
The format of the audio data. If a non-NULL value is specified for this parameter, then the format plug-in for this format type is invoked; otherwise, the default plug-in is used.

Usage Notes

None.

Pragmas

None.

Exceptions

ORDAudioExceptions.AUDIO_PLUGIN_EXCEPTION

This exception is raised if you call the getProperties( ) method and the audio plug-in raises an exception.

ORDSourceExceptions.EMPTY_SOURCE

This exception is raised when the value of the source.local attribute is 1 or 0 (TRUE), but the value of the source.localData attribute is NULL.

Examples

Get the property information for known audio attributes:

```
DECLARE
  aud_attrib CLOB;
  ctx RAW(64) :=NULL;
  aud_data BLOB;
  aud_format VARCHAR2(160) := NULL;
BEGIN
  SELECT aud, attributes INTO aud_data, aud_attrib FROM taud WHERE N =1 FOR UPDATE;
  ORDSYS.ORDAudio.getProperties(ctx,aud_data,aud_attrib,aud_format);
  DBMS_OUTPUT.put_line('Size of XML Annotations: ' ||
                 TO_CHAR(DBMS_LOB.GETLENGTH(aud_attrib)));
  UPDATE taud SET attributes=aud_attrib WHERE N=1;
  COMMIT;
EXCEPTION
  WHEN OTHERS THEN
  RAISE;
END;
/
```

getProperties( ) (all attributes) for BLOBs

Format

```
getProperties(ctx                 IN OUT RAW,
              audioBlob           IN BLOB,
              attributes          IN OUT NOCOPY CLOB,
              mimeType            OUT VARCHAR2,
              format              IN OUT VARCHAR2
              encoding            OUT VARCHAR2,
              numberOfChannels    OUT INTEGER,
              samplingRate        OUT INTEGER,
              sampleSize          OUT INTEGER,
              compressionType     OUT VARCHAR2,
              audioDuration       OUT INTEGER);
```

Description

Reads the audio BLOB data to get the values of the media attributes for supported formats, and then stores them in the input CLOB and returns them as explicit parameters. This method gets the properties for the following attributes of the audio data: duration, MIME type, compression type, format, encoding type, number of channels, sampling rate, and sample size. It populates the CLOB with an extensive set of format and application properties in XML form.

Parameters

ctx
The format plug-in context information.

audioBlob
The audio data represented as a BLOB.

attributes
The CLOB to hold the XML attribute information generated by the getProperties( ) method. This CLOB is populated with an extensive set of format and application properties of the audio BLOB data in XML form.

mimeType
The MIME type of the audio data.

format
The format of the audio data. If a non-NULL value is specified, then the format plug-in for this format type is invoked. If not specified, the derived format value is returned.

encoding
The encoding type of the audio data.

numberOfChannels
The number of channels in the audio data.

samplingRate
The sampling rate in samples per second at which the audio data was recorded.

sampleSize
The sample width or number of samples of audio in the data.

compressionType
The compression type of the audio data.

audioDuration
The total time required to play the audio data.

Usage Notes

If a property cannot be extracted from the media source, then the respective parameter is set to NULL.

Pragmas

None.

Exceptions

ORDAudioExceptions.AUDIO_PLUGIN_EXCEPTION

This exception is raised if you call the getProperties( ) method and the audio plug-in raises an exception.

ORDSourceExceptions.EMPTY_SOURCE

This exception is raised when the value of the source.local attribute is 1 or 0 (TRUE), but the value of the source.localData attribute is NULL.

Examples

Get the property information for known audio attributes:

```
DECLARE
  aud_attrib        CLOB;
  ctx               RAW(64) := NULL;
  aud_data          BLOB;
  mimeType          VARCHAR2(80);
  format            VARCHAR2(32) := NULL;
  encoding          VARCHAR2(160);
  numberOfChannels  NUMBER;
  samplingRate      NUMBER;
  sampleSize        NUMBER;
  compressionType   VARCHAR2(160);
  audioDuration     NUMBER;
BEGIN
  SELECT aud, attributes, mimetype, format, encoding, numberofchannels, samplingrate,
    samplesize, compressiontype, audioduration INTO aud_data, aud_attrib, mimeType,
    format, encoding, numberOfChannels, samplingRate, sampleSize, compressionType,
    audioDuration FROM taud WHERE N = 1 FOR UPDATE;

  ORDSYS.ORDAudio.getProperties(ctx, aud_data, aud_attrib, mimeType, format, encoding,
              numberOfChannels, samplingRate, sampleSize, compressionType, audioDuration);

  DBMS_OUTPUT.put_line('Size of XML Annotations ' ||
                    TO_CHAR(DBMS_LOB.GETLENGTH(aud_attrib)));
  DBMS_OUTPUT.put_line('mimeType: ' || mimeType );
  DBMS_OUTPUT.put_line('format: ' || format );
  DBMS_OUTPUT.put_line('encoding: ' || encoding );
  DBMS_OUTPUT.put_line('numberOfChannels: ' || numberOfChannels );
  DBMS_OUTPUT.put_line('samplingRate: ' || samplingRate );
  DBMS_OUTPUT.put_line('sampleSize: ' || sampleSize );
  DBMS_OUTPUT.put_line('compressionType: ' || compressionType );
  DBMS_OUTPUT.put_line('audioDuration: ' || audioDuration );
  UPDATE taud SET
               aud=aud_data,
               attributes=aud_attrib,
               mimetype=mimeType,
               format=format,
               encoding=encoding,
```

```
              numberofchannels=numberOfChannels,
              samplingrate=samplingRate,
              samplesize=sampleSize,
              compressiontype=compressionType,
              audioduration=audioDuration
            WHERE n=1;
  COMMIT;
EXCEPTION
  WHEN OTHERS THEN
  RAISE;
END;
/
```

getProperties( ) for BFILEs

Format

```
getProperties(ctx            IN OUT RAW,
              audioBfile     IN OUT NOCOPY BFILE,
              attributes     IN OUT NOCOPY CLOB,
              format         IN VARCHAR2);
```

Description

Reads the audio BFILE data to get the values of the media attributes for supported formats, and then stores them in the input CLOB. This method populates the CLOB with an extensive set of format and application properties in XML form.

Parameters

ctx
The format plug-in context information.

audioBfile
The audio data represented as a BFILE.

attributes
The CLOB to hold the XML attribute information generated by the getProperties( ) method. This CLOB is populated with an extensive set of format and application properties of the audio BFILE data in XML form.

format
The format of the audio data. If a non-NULL value is specified for this parameter, then the format plug-in for this format type is invoked.

Usage Notes

None.

Pragmas

None.

Exceptions

ORDAudioExceptions.AUDIO_PLUGIN_EXCEPTION

This exception is raised if you call the getProperties( ) method and the audio plug-in raises an exception.

ORDSourceExceptions.EMPTY_SOURCE

This exception is raised when the value of the source.local attribute is 1 or 0 (TRUE), but the value of the source.localData attribute is NULL.

Examples

Get the property information for known audio attributes:

```
DECLARE
  aud_attrib CLOB;
  ctx RAW(64) :=NULL;
  aud_data BFILE := BFILENAME('AUDIODIR','testaud.dat');
  aud_format VARCHAR2(160) := NULL;
BEGIN
  DBMS_LOB.CREATETEMPORARY(aud_attrib, FALSE, DBMS_LOB.CALL);
  ORDSYS.ORDAudio.getProperties(ctx, aud_data, aud_attrib, aud_format);

  DBMS_OUTPUT.put_line('Size of XML Annotations ' ||
                  TO_CHAR(DBMS_LOB.GETLENGTH(aud_attrib)));
EXCEPTION
  WHEN OTHERS THEN
  RAISE;
END;
/
```

getProperties( ) (all attributes) for BFILEs

Format

```
getProperties(ctx                IN OUT RAW,
              audioBfile         IN OUT NOCOPY BFILE,
              attributes         IN OUT NOCOPY CLOB,
              mimeType           OUT VARCHAR2,
              format             IN OUT VARCHAR2
              encoding           OUT VARCHAR2,
              numberOfChannels   OUT INTEGER,
              samplingRate       OUT INTEGER,
              sampleSize         OUT INTEGER,
              compressionType    OUT VARCHAR2,
              audioDuration      OUT INTEGER);
```

Description

Reads the audio BFILE data to get the values of the media attributes for supported formats, and then stores them in the input CLOB and returns them as explicit parameters. This method gets the properties for the following attributes of the audio data: duration, MIME type, compression type, format, encoding type, number of channels, sampling rate, and sample size. It populates the CLOB with an extensive set of format and application properties in XML form.

Parameters

ctx
The format plug-in context information.

audioBfile
The audio data represented as a BFILE.

attributes
The CLOB to hold the XML attribute information generated by the getProperties( ) method. This CLOB is populated with an extensive set of format and application properties of the audio BFILE data in XML form.

mimeType
The MIME type of the audio data.

format
The format of the audio data. If a non-NULL value is specified, then the format plug-in for this format type is invoked. If not specified, the default plug-in is used and the derived format value is returned.

encoding
The encoding type of the audio data.

numberOfChannels
The number of channels in the audio data.

samplingRate
The sampling rate in samples per second at which the audio data was recorded.

sampleSize
The sample width or number of samples of audio in the data.

compressionType
The compression type of the audio data.

audioDuration
The total time required to play the audio data.

Usage Notes

If a property cannot be extracted from the media source, then the respective parameter is set to NULL.

Pragmas

None.

Exceptions

ORDAudioExceptions.AUDIO_PLUGIN_EXCEPTION

This exception is raised if you call the getProperties( ) method and the audio plug-in raises an exception.

ORDSourceExceptions.EMPTY_SOURCE

This exception is raised when the value of the source.local attribute is 1 or 0 (TRUE), but the value of the source.localData attribute is NULL.

Examples

Get the property information for known audio attributes:

```
DECLARE
  aud_attrib        CLOB;
  ctx               RAW(64) :=NULL;
  data              BFILE := BFILENAME('AUDIODIR','testaud.dat');
  mimeType          VARCHAR2(80);
  format            VARCHAR2(32) := NULL;
  encoding          VARCHAR2(160);
  numberOfChannels  NUMBER;
  samplingRate      NUMBER;
  sampleSize        NUMBER;
  compressionType   VARCHAR2(160);
  audioDuration     NUMBER;
BEGIN
  DBMS_LOB.CREATETEMPORARY(aud_attrib, FALSE, DBMS_LOB.CALL);

  ORDSYS.ORDAudio.getProperties(ctx, data, aud_attrib, mimeType, format, encoding,
              numberOfChannels, samplingRate, sampleSize, compressionType,
 audioDuration);

  DBMS_OUTPUT.put_line('Size of XML Annotations ' ||
                  TO_CHAR(DBMS_LOB.GETLENGTH(aud_attrib)));
  DBMS_OUTPUT.put_line('mimeType: ' || mimeType );
  DBMS_OUTPUT.put_line('format: ' || format );
  DBMS_OUTPUT.put_line('encoding: ' || encoding );
  DBMS_OUTPUT.put_line('numberOfChannels: ' || numberOfChannels );
  DBMS_OUTPUT.put_line('samplingRate: ' || samplingRate );
  DBMS_OUTPUT.put_line('sampleSize: ' || sampleSize );
  DBMS_OUTPUT.put_line('compressionType: ' || compressionType );
  DBMS_OUTPUT.put_line('audioDuration: ' || audioDuration );
EXCEPTION
  WHEN OTHERS THEN
  RAISE;
END;
/
```

Static Methods Unique to the ORDDoc Object Type Relational Interface

This section presents reference information on the *inter*Media static methods unique to the ORDDoc relational interface.

The relational interface adds *inter*Media support to audio, image, video, and other heterogeneous media data stored in BLOBs and BFILEs rather than in the ORDDoc object type. The following interface is defined in the orddspec.sql file:

```
.
.
.
  -- Static Methods for the relational interface
  STATIC PROCEDURE export(ctx             IN OUT RAW,
                          local_data      IN BLOB,
                          source_type     IN VARCHAR2,
                          source_location IN VARCHAR2,
                          source_name     IN VARCHAR2),
  --
  STATIC PROCEDURE importFrom(ctx             IN OUT RAW,
                              local_data      IN OUT NOCOPY BLOB,
                              source_type     IN VARCHAR2,
                              source_location IN VARCHAR2,
                              source_name     IN VARCHAR2),
  --
  STATIC PROCEDURE importFrom(ctx             IN OUT RAW,
                              local_data      IN OUT NOCOPY BLOB,
                              source_type     IN VARCHAR2,
                              source_location IN VARCHAR2,
                              source_name     IN VARCHAR2,
                              format          OUT VARCHAR2,
                              mime_type       OUT VARCHAR2),
  --
  STATIC PROCEDURE getProperties(ctx            IN OUT RAW,
                                 docBlob        IN BLOB,
                                 attributes     IN OUT NOCOPY CLOB,
                                 format         IN VARCHAR2),
  --
  STATIC PROCEDURE getProperties(ctx              IN OUT RAW,
                                 docBlob          IN BLOB,
                                 attributes       IN OUT NOCOPY CLOB,
                                 mimeType         OUT VARCHAR2,
                                 format           IN OUT VARCHAR2,
```

```
                                contentLength    OUT INTEGER),
 --
  STATIC PROCEDURE getProperties(ctx            IN OUT RAW,
                                 docBfile       IN OUT NOCOPY BFILE,
                                 attributes     IN OUT NOCOPY CLOB,
                                 format         IN VARCHAR2),
  --
  STATIC PROCEDURE getProperties(ctx               IN OUT RAW,
                                 docBfile          IN OUT NOCOPY BFILE,
                                 attributes        IN OUT NOCOPY CLOB,
                                 mimeType          OUT VARCHAR2,
                                 format            IN OUT VARCHAR2,
                                 contentLength     OUT INTEGER),
.
.
.
```

Example Media Table Definition

The methods described in this section show examples based on a test document table TDOC. Refer to the TDOC table definition that follows when reading through the examples:

TDOC Table Definition

```
CREATE TABLE tdoc(n             NUMBER,
                  document      BLOB,
                  attributes    CLOB,
                  mimetype      VARCHAR2(80),
                  format        VARCHAR2(80),
                  contentlength INTEGER)
STORAGE (INITIAL 100K NEXT 100K PCTINCREASE 0);

INSERT INTO tdoc VALUES(1, EMPTY_BLOB(), EMPTY_CLOB(), NULL, NULL, NULL);
INSERT INTO tdoc VALUES(2, EMPTY_BLOB(), EMPTY_CLOB(), NULL, NULL, NULL);
INSERT INTO tdoc VALUES(3, EMPTY_BLOB(), EMPTY_CLOB(), NULL, NULL, NULL);
INSERT INTO tdoc VALUES(4, EMPTY_BLOB(), EMPTY_CLOB(), NULL, NULL, NULL);
COMMIT;
```

getProperties( ) for BLOBs

Format

```
getProperties(ctx          IN OUT RAW,
              docBlob      IN BLOB,
              attributes   IN OUT NOCOPY CLOB,
              format       IN VARCHAR2);
```

Description

Reads the document BLOB data to get the values of the media attributes, and then stores them in the input CLOB. This method populates the CLOB with an extensive set of format and application properties in XML form.

Parameters

ctx
The format plug-in context information.

docBlob
The document data represented as a BLOB.

attributes
The CLOB to hold the XML attribute information generated by the getProperties( ) method. This CLOB is populated with an extensive set of format and application properties of the document BLOB data in XML form.

format
The format of the document data. If a non-NULL value is specified, then the format plug-in for this format type is invoked.

Usage Notes

None.

Pragmas

None.

Exceptions

ORDDocExceptions.DOC_PLUGIN_EXCEPTION

This exception is raised if you call the getProperties( ) method and the document plug-in raises an exception.

ORDSourceExceptions.EMPTY_SOURCE

This exception is raised when the value of the source.local attribute is 1 or 0 (TRUE), but the value of the source.localData attribute is NULL.

Examples

Get the property information for known document attributes:

```
DECLARE
  doc_attrib CLOB;
  ctx RAW(64) :=NULL;
  doc_data BLOB;
  doc_format VARCHAR2(160) := NULL;

BEGIN
    SELECT document,attributes INTO doc_data,doc_attrib FROM tdoc WHERE N = 1 FOR UPDATE;
   ORDSYS.ORDDoc.getProperties(ctx, doc_data, doc_attrib, doc_format);

   DBMS_OUTPUT.put_line('Size of XML Annotations ' ||
                   TO_CHAR(DBMS_LOB.GETLENGTH(doc_attrib)));
   UPDATE tdoc SET document=doc_data, attributes=doc_attrib WHERE N=1;
   COMMIT;
   EXCEPTION
      WHEN OTHERS THEN
      RAISE;
END;
/
```

getProperties( ) (all attributes) for BLOBs

Format

```
getProperties(ctx            IN OUT RAW,
              docBlob        IN BLOB,
              attributes     IN OUT NOCOPY CLOB,
              mimeType       OUT VARCHAR2,
              format         IN OUT VARCHAR2,
              contentLength  OUT INTEGER);
```

Description

Reads the document BLOB data to get the values of the media attributes, and then stores them in the input CLOB and returns them as explicit parameters. This method gets the properties for the following attributes of the document data: MIME type, content length, and format. It populates the CLOB with an extensive set of format and application properties in XML form.

Parameters

ctx
The format plug-in context information.

docBlob
The document data represented as a BLOB.

attributes
The CLOB to hold the XML attribute information generated by the getProperties( ) method. This CLOB is populated with an extensive set of format and application properties of the document BLOB data in XML form.

mimeType
The MIME type of the document data.

format
The format of the document data. If a non-NULL value is specified, then the format plug-in for this format type is invoked.

contentLength
The length in bytes of the content.

Usage Notes

If a property cannot be extracted from the media source, then the respective parameter is set to NULL.

Pragmas

None.

Exceptions

ORDDocExceptions.DOC_PLUGIN_EXCEPTION

This exception is raised if you call the getProperties( ) method and the document plug-in raises an exception.

ORDSourceExceptions.EMPTY_SOURCE

This exception is raised when the value of the source.local attribute is 1 or 0 (TRUE), but the value of the source.localData attribute is NULL.

Examples

Get the property information for known document attributes:

```
DECLARE
  doc_attrib        CLOB;
  ctx               RAW(64) :=NULL;
  doc_data          BLOB;
  doc_mimeType      VARCHAR2(80);
  doc_format        VARCHAR2(32):=NULL;
  doc_contentLength NUMBER;
BEGIN
   SELECT document, attributes, mimetype, format, contentlength INTO doc_data,
     doc_attrib, doc_mimeType, doc_format, doc_contentLength FROM tdoc
     WHERE N = 1 FOR UPDATE;

   ORDSYS.ORDDoc.getProperties(ctx, doc_data, doc_attrib,
                 doc_mimeType, doc_format, doc_contentLength);

   DBMS_OUTPUT.put_line('Size of XML Annotations ' ||
                  TO_CHAR(DBMS_LOB.GETLENGTH(doc_attrib)));
   DBMS_OUTPUT.put_line('mimeType: ' || doc_mimeType );
   DBMS_OUTPUT.put_line('format: ' || doc_format );
   DBMS_OUTPUT.put_line('contentLength: ' || doc_contentLength );
```

```
   UPDATE tdoc SET
     document=doc_data,
     attributes=doc_attrib,
     mimetype=doc_mimeType,
     format=doc_format,
     contentlength=doc_contentLength
    WHERE N=1;
   COMMIT;
EXCEPTION
   WHEN OTHERS THEN
   RAISE;
END;
/
```

getProperties( ) for BFILEs

Format

```
getProperties(ctx          IN OUT RAW,
              docBfile     IN OUT NOCOPY BFILE,
              attributes   IN OUT NOCOPY CLOB,
              format       IN VARCHAR2);
```

Description

Reads the document BFILE data to get the values of the media attributes, and then stores them in the input CLOB. It populates the CLOB with an extensive set of format and application properties in XML form.

Parameters

ctx
The format plug-in context information.

docBfile
The document data represented as a BFILE.

attributes
The CLOB to hold the XML attribute information generated by the getProperties( ) method. This CLOB is populated with an extensive set of format and application properties of the document BFILE data in XML form.

format
The format of the document data. If a non-NULL value is specified, then the format plug-in for this format type is invoked.

Usage Notes

None.

Pragmas

None.

Exceptions

ORDDocExceptions.DOC_PLUGIN_EXCEPTION

This exception is raised if you call the getProperties( ) method and the document plug-in raises an exception.

ORDSourceExceptions.EMPTY_SOURCE

This exception is raised when the value of the source.local attribute is 1 or 0 (TRUE), but the value of the source.localData attribute is NULL.

Examples

Get the property information for known document attributes:

```
DECLARE
  doc_attrib CLOB;
  ctx RAW(64) :=NULL;
  doc_data BFILE := BFILENAME('DOCDIR','testvid.dat');
  doc_format VARCHAR2(160) := NULL;
BEGIN
   DBMS_LOB.CREATETEMPORARY(doc_attrib, FALSE, DBMS_LOB.CALL);
   ORDSYS.ORDDoc.getProperties(ctx, doc_data, doc_attrib, doc_format);

   DBMS_OUTPUT.put_line('Size of XML Annotations ' ||
                  TO_CHAR(DBMS_LOB.GETLENGTH(doc_attrib)));
EXCEPTION
   WHEN OTHERS THEN
   RAISE;
END;
/
```

getProperties( ) (all attributes) for BFILEs

Format

```
getProperties(ctx              IN OUT RAW,
              docBfile         IN OUT NOCOPY BFILE,
              attributes       IN OUT NOCOPY CLOB,
              mimeType         OUT VARCHAR2,
              format           IN OUT VARCHAR2,
              contentLength    OUT INTEGER);
```

Description

Reads the document BFILE data to get the values of the media attributes for supported formats, and then stores them in the input CLOB and returns them as explicit parameters. This method gets the properties for the following attributes of the document data: MIME type, content length, and format. It populates the CLOB with an extensive set of format and application properties in XML form.

Parameters

ctx
The format plug-in context information.

docBfile
The document data represented as a BFILE.

attributes
The CLOB to hold the XML attribute information generated by the getProperties( ) method. This CLOB is populated with an extensive set of format and application properties of the document BFILE data in XML form.

mimeType
The MIME type of the document data.

format
The format of the document data. If a non-NULL value is specified, then the format plug-in for this format type is invoked. If not specified, the derived format is returned.

contentLength
The length in bytes of the content.

Usage Notes

If a property cannot be extracted from the media source, then the respective parameter is set to NULL.

Pragmas

None.

Exceptions

ORDDocExceptions.DOC_PLUGIN_EXCEPTION

This exception is raised if you call the getProperties( ) method and the document plug-in raises an exception.

ORDSourceExceptions.EMPTY_SOURCE

This exception is raised when the value of the source.local attribute is 1 or 0 (TRUE), but the value of the source.localData attribute is NULL.

Examples

Get the property information for known document attributes:

```
DECLARE
  doc_attrib        CLOB;
  ctx               RAW(64) :=NULL;
  doc_data          BFILE := BFILENAME('DOCDIR','testimg.dat');
  doc_mimeType      VARCHAR2(80);
  doc_format        VARCHAR2(32) := NULL;
  doc_contentLength NUMBER;
BEGIN
   DBMS_LOB.CREATETEMPORARY(doc_attrib, FALSE, DBMS_LOB.CALL);
   ORDSYS.ORDDoc.getProperties(ctx, doc_data, doc_attrib,
                doc_mimeType, doc_format, doc_contentLength);
   DBMS_OUTPUT.put_line('Size of XML Annotations ' || TO_CHAR(DBMS_LOB.GETLENGTH(doc_attrib)));
   DBMS_OUTPUT.put_line('mimeType: ' || doc_mimeType );
   DBMS_OUTPUT.put_line('format: ' || doc_format );
   DBMS_OUTPUT.put_line('contentLength: ' || doc_contentLength );
EXCEPTION
   WHEN OTHERS THEN
   RAISE;
END;
/
```

Static Methods Unique to the ORDImage Object Type Relational Interface

This section presents reference information on the *inter*Media static methods unique to the ORDImage relational interface.

The relational interface adds *inter*Media support to image data stored in BLOBs and BFILEs rather than in the ORDImage object type. The following interface is defined in the ordispec.sql file:

```
.
.
.
  -- Static Methods for the relational interface
  STATIC PROCEDURE export(ctx             IN OUT RAW,
                          local_data      IN BLOB,
                          source_type     IN VARCHAR2,
                          source_location IN VARCHAR2,
                          source_name     IN VARCHAR2),
  --
  STATIC PROCEDURE importFrom(ctx             IN OUT RAW,
                              local_data      IN OUT NOCOPY BLOB,
                              source_type     IN VARCHAR2,
                              source_location IN VARCHAR2,
                              source_name     IN VARCHAR2),
  --
  STATIC PROCEDURE importFrom(ctx             IN OUT RAW,
                              local_data      IN OUT NOCOPY BLOB,
                              source_type     IN VARCHAR2,
                              source_location IN VARCHAR2,
                              source_name     IN VARCHAR2,
                              format          OUT VARCHAR2,
                              mime_type       OUT VARCHAR2),
  --
  STATIC PROCEDURE getProperties(imageBlob         IN BLOB,
                                 attributes        IN OUT NOCOPY CLOB,
                                 mimeType          OUT VARCHAR2,
                                 width             OUT INTEGER,
                                 height            OUT INTEGER,
                                 fileFormat        OUT VARCHAR2,
                                 contentFormat     OUT VARCHAR2,
                                 compressionFormat OUT VARCHAR2,
                                 contentLength     OUT INTEGER),
  --
```

```
  STATIC PROCEDURE getProperties(imageBlob         IN BLOB,
                                 attributes        IN OUT NOCOPY CLOB),
  --
  STATIC PROCEDURE getProperties(imageBfile       IN OUT NOCOPY BFILE,
                                 attributes        IN OUT NOCOPY CLOB,
                                 mimeType          OUT VARCHAR2,
                                 width             OUT INTEGER,
                                 height            OUT INTEGER,
                                 fileFormat        OUT VARCHAR2,
                                 contentFormat     OUT VARCHAR2,
                                 compressionFormat OUT VARCHAR2,
                                 contentLength     OUT INTEGER),
  --
  STATIC PROCEDURE getProperties(imageBfile IN OUT NOCOPY BFILE,
                                 attributes IN OUT NOCOPY CLOB),
  --
  STATIC PROCEDURE process(imageBlob IN OUT NOCOPY BLOB,
                           command   IN VARCHAR2),
  --
  STATIC PROCEDURE processCopy(imageBlob IN OUT NOCOPY BLOB,
                               command   IN VARCHAR2,
                               dest      IN OUT NOCOPY BLOB),
  --
  STATIC PROCEDURE processCopy(imageBfile IN OUT BFILE,
                               command    IN VARCHAR2,
                               dest       IN OUT NOCOPY BLOB),
.
.
.
```

Example Image Table Definition

The methods described in this section show examples based on a test image table TIMG. Refer to the TIMG table definition that follows when reading through the examples:

TIMG Table Definition

```
CREATE TABLE timg(n NUMBER,
                  img BLOB,
                  attributes CLOB,
                  mimetype VARCHAR2(4000),
                  width INTEGER,
                  height INTEGER,
                  fileformat VARCHAR2(4000),
```

```
                  contentformat VARCHAR2(4000),
                  compressionformat VARCHAR2(4000),
                  contentlength INTEGER)
STORAGE (INITIAL 100K NEXT 100K PCTINCREASE 0);

INSERT INTO timg VALUES(1, EMPTY_BLOB(), EMPTY_CLOB(), NULL,
                        NULL, NULL, NULL, NULL, NULL, NULL);
INSERT INTO timg VALUES(2, EMPTY_BLOB(), EMPTY_CLOB(), NULL,
                        NULL, NULL, NULL, NULL, NULL, NULL);
COMMIT;
```

getProperties( ) for BLOBs

Format

```
getProperties(imageBlob      IN BLOB,
              attributes     IN OUT NOCOPY CLOB);
```

Description

Reads the image BLOB data to get the values of the media attributes for supported formats, and then stores them in the input CLOB. This method populates the CLOB with a set of format properties in XML form.

Parameters

imageBlob
The image data represented as a BLOB.

attributes
The CLOB to hold the XML attribute information generated by the getProperties( ) method. This CLOB is populated with a set of format properties of the image BLOB data in XML form.

Usage Notes

None.

Pragmas

None.

Exceptions

ORDImageExceptions.NULL_CONTENT

This exception is raised when the imageBlob parameter is NULL.

Examples

Get the property information for known image attributes:

```
DECLARE
  img_attrib CLOB;
  img_data BLOB;
```

```
BEGIN
   SELECT img, attributes INTO img_data, img_attrib FROM timg WHERE N = 1 FOR UPDATE;
   ORDSYS.ORDImage.getProperties(img_data, img_attrib);

   DBMS_OUTPUT.put_line('Size of XML Annotations ' ||
                  TO_CHAR(DBMS_LOB.GETLENGTH(img_attrib)));
   UPDATE timg SET img=img_data, attributes=img_attrib WHERE N=1;
   COMMIT;
EXCEPTION
   WHEN OTHERS THEN
   RAISE;
END;
/
```

getProperties( ) (all attributes) for BLOBs

Format

```
getProperties(imageBlob          IN BLOB,
              attributes         IN OUT NOCOPY CLOB,
              mimeType           OUT VARCHAR2,
              width              OUT INTEGER,
              height             OUT INTEGER,
              fileFormat         OUT VARCHAR2,
              contentFormat      OUT VARCHAR2,
              compressionFormat  OUT VARCHAR2,
              contentLength      OUT INTEGER);
```

Description

Reads the image BLOB data to get the values of the media attributes for supported formats, and then stores them in the input CLOB and returns them as explicit parameters. This method gets the properties for the following attributes of the image data: MIME type, width, height, file format, content format, compression format, and content length. It populates the CLOB with a set of format properties in XML form.

Parameters

imageBlob
The image data represented as a BLOB.

attributes
The CLOB to hold the XML attribute information generated by the getProperties( ) method. This CLOB is populated with a set of format properties of the image BLOB data in XML form.

mimeType
The MIME type of the image data.

width
The width of the image in pixels.

height
The height of the image in pixels.

fileFormat
The format of the image data.

contentFormat
The type of image (monochrome, and so forth).

compressionFormat
The compression algorithm used on the image data.

contentLength
The size of the image file on disk, in bytes.

Usage Notes

If a property cannot be extracted from the media source, then the respective parameter is set to NULL.

Pragmas

None.

Exceptions

ORDImageExceptions.NULL_CONTENT

This exception is raised when the imageBlob parameter is NULL.

Examples

Get the property information for known image attributes:

```
DECLARE
  img_data             BLOB;
  img_attrib        CLOB;
  mimeType          VARCHAR2(4000);
  width             NUMBER;
  height            NUMBER;
  fileFormat        VARCHAR2(32);
  contentFormat     VARCHAR2(4000);
  compressionFormat VARCHAR2(4000);
```

```
  contentLength     NUMBER;
BEGIN
   SELECT img, attributes, mimetype, width, height, fileformat, contentformat,
     compressionformat, contentlength INTO img_data, img_attrib, mimeType, width,
     height, fileFormat, contentFormat, compressionFormat, contentLength
     FROM timg WHERE N = 1 FOR UPDATE;

  ORDSYS.ORDImage.getProperties(img_data, img_attrib,
        mimeType, width, height, fileFormat,
        contentFormat, compressionFormat, contentLength);

   DBMS_OUTPUT.put_line('Size of XML Annotations ' ||
                 TO_CHAR(DBMS_LOB.GETLENGTH(img_attrib)));
   DBMS_OUTPUT.put_line('mimeType: ' || mimeType );
   DBMS_OUTPUT.put_line('width: ' || width );
   DBMS_OUTPUT.put_line('height: ' || height );
   DBMS_OUTPUT.put_line('fileFormat: ' || fileFormat );
   DBMS_OUTPUT.put_line('contentFormat: ' || contentFormat );
   DBMS_OUTPUT.put_line('compressionFormat: ' || compressionFormat );
   DBMS_OUTPUT.put_line('contentLength: ' || contentLength );
   UPDATE timg SET
       img=img_data,
       attributes=img_attrib,
       mimetype=mimeType,
       width=width,
       height=height,
       fileformat=fileFormat,
       contentformat=contentFormat,
       compressionformat=compressionFormat,
       contentlength=contentLength
    WHERE N=1;
   COMMIT;
EXCEPTION
   WHEN OTHERS THEN
   RAISE;
END;
/
```

getProperties( ) for BFILEs

Format

```
getProperties(imageBfile      IN OUT NOCOPY BFILE,
              attributes      IN OUT NOCOPY CLOB);
```

Description

Reads the image BFILE data to get the values of the media attributes for supported formats, and then stores them in the input CLOB. This method populates the CLOB with a set of format properties in XML form.

Parameters

imageBfile
The image data represented as a BFILE.

attributes
The CLOB to hold the XML attribute information generated by the getProperties( ) method. This CLOB is populated with a set of format properties of the image BFILE data in XML form.

Usage Notes

None.

Pragmas

None.

Exceptions

ORDImageExceptions.NULL_CONTENT

This exception is raised when the imageBfile parameter is NULL.

Examples

Get the property information for known image attributes:

```
DECLARE
  img_attrib CLOB;
  data BFILE := BFILENAME('IMAGEDIR','testimg.dat');
```

```
BEGIN
   DBMS_LOB.CREATETEMPORARY(img_attrib, FALSE, DBMS_LOB.CALL);
   ORDSYS.ORDImage.getProperties(data, img_attrib);

   DBMS_OUTPUT.put_line('Size of XML Annotations ' ||
                   TO_CHAR(DBMS_LOB.GETLENGTH(img_attrib)));
EXCEPTION
   WHEN OTHERS THEN
   RAISE;
END;
/
```

getProperties( ) (all attributes) for BFILEs

Format

```
getProperties(imageBfile          IN OUT NOCOPY BFILE,
              attributes          IN OUT NOCOPY CLOB,
              mimeType            OUT VARCHAR2,
              width               OUT INTEGER,
              height              OUT INTEGER,
              fileFormat          OUT VARCHAR2,
              contentFormat       OUT VARCHAR2,
              compressionFormat   OUT VARCHAR2,
              contentLength       OUT INTEGER);
```

Description

Reads the image BFILE data to get the values of the media attributes for supported formats, and then stores them in the input CLOB and returns them as explicit parameters. This method gets the properties for the following attributes of the image data: MIME type, width, height, file format, content format, compression format, and content length. It populates the CLOB with a set of format properties in XML form.

Parameters

imageBfile
The image data represented as a BFILE.

attributes
The CLOB to hold the XML attribute information generated by the getProperties( ) method. This CLOB is populated with a set of format properties of the image BFILE data in XML form.

mimeType
The MIME type of the image data.

width
The width of the image in pixels.

height
The height of the image in pixels.

fileFormat
The format of the image data.

contentFormat
The type of image (monochrome, and so forth).

compressionFormat
The compression algorithm used on the image data.

contentLength
The size of the image file on disk, in bytes.

Usage Notes

If a property cannot be extracted from the media source, then the respective parameter is set to NULL.

Pragmas

None.

Exceptions

ORDImageExceptions.NULL_CONTENT

This exception is raised when the imageBfile parameter is NULL.

Examples

Get the property information for known image attributes:

```
DECLARE
  img_data          BFILE := BFILENAME('IMAGEDIR','testimg.dat');
  img_attrib        CLOB;
  mimeType          VARCHAR2(80);
  width             NUMBER;
  height            NUMBER;
  fileFormat        VARCHAR2(32);
  contentFormat     VARCHAR2(4000);
  compressionFormat VARCHAR2(4000);
  contentLength     NUMBER;
BEGIN
   DBMS_LOB.CREATETEMPORARY(img_attrib, FALSE, DBMS_LOB.CALL);
```

```
  ORDSYS.ORDImage.getProperties(img_data, img_attrib,
         mimeType, width, height, fileFormat,
         contentFormat, compressionFormat, contentLength);

   DBMS_OUTPUT.put_line('Size of XML Annotations ' ||
                  TO_CHAR(DBMS_LOB.GETLENGTH(img_attrib)));
   DBMS_OUTPUT.put_line('mimeType: ' || mimeType );
   DBMS_OUTPUT.put_line('width: ' || width );
   DBMS_OUTPUT.put_line('height: ' || height );
   DBMS_OUTPUT.put_line('fileFormat: ' || fileFormat );
   DBMS_OUTPUT.put_line('contentFormat: ' || contentFormat );
   DBMS_OUTPUT.put_line('compressionFormat: ' || compressionFormat );
   DBMS_OUTPUT.put_line('contentLength: ' || contentLength );
EXCEPTION
   WHEN OTHERS THEN
   RAISE;
END;
/
```

process( )

Format

```
process(imageBlob IN OUT NOCOPY BLOB,
        command IN VARCHAR2);
```

Description

Performs one or more image processing operations on a BLOB, writing the image back onto itself.

Parameters

imageBlob
The image data represented as a BLOB.

command
A list of image processing operations to perform on the image.

Usage Notes

You can change one or more of the image attributes shown in Table 6–1. Table 6–2 shows additional changes that can be made only to raw pixel and foreign images.

See Appendix D for more information on process( ) operators.

The process( ) method changes image attributes, therefore if you are storing image attributes, you should call the getProperties( ) method after calling the process( ) method.

Pragmas

None.

Exceptions

ORDImageExceptions.DATA_NOT_LOCAL

This exception is raised if you call the process( ) method and the imageBlob parameter is not initialized.

Examples

Example 1: Change the image in the image_data BLOB to use higher quality JPEG compression and double the length of the image along the X-axis:

```
ORDSYS.ORDImage.process(
image_data,'compressionFormat=JPEG,compressionQuality=MAXCOMPRATIO, xScale="2.0"');
```

Note that changing the length on only one axis (for example, xScale=2.0) does not affect the length on the other axis, and would result in image distortion. Also, only the xScale and yScale parameters can be combined in a single scale operation. Any other combinations of scale operators result in an error.

Example 2: Create at most a 32-by-32 pixel thumbnail image, preserving the original aspect ratio. The maxScale and fixedScale operators are especially useful for creating thumbnail images from various-sized originals:

```
ORDSYS.ORDImage.process(image_data, 'maxScale=32 32');
```

Example 3: Convert the image to TIFF:

```
DECLARE
img_attrib CLOB;
image_data BLOB;
BEGIN
   SELECT img, attributes INTO image_data, img_attrib FROM timg
     WHERE N = 1 FOR UPDATE;
   ORDSYS.ORDImage.process(image_data, 'fileFormat=TIFF');
   ORDSYS.ORDImage.getProperties(image_data, img_attrib);
   UPDATE timg SET img = image_data, attributes=img_attrib WHERE N = 1;
   COMMIT;
EXCEPTION
   WHEN OTHERS THEN
   RAISE;
END;
/
```

processCopy( ) for BLOBs

Format

```
processCopy(imageBlob IN BLOB,
            command  IN VARCHAR2,
            dest     IN OUT NOCOPY  BLOB);
```

Description

Copies an image stored internally or externally to another image stored internally in the source.localData attribute (of the embedded ORDSource object) and performs one or more image processing operations on the copy.

Parameters

imageBlob
The source image data represented as a BLOB.

command
A list of image processing changes to make for the image in the new copy.

dest
The destination of the new image.

Usage Notes

See Table 6–1, " Image Processing Operators" and Table 6–2, " Additional Image Processing Operators for Raw Pixel and Foreign Images".

You cannot specify the same BLOB as both the source and destination.

Calling this method processes the image into the destination BLOB from any source BLOB.

The processCopy( ) method changes image attributes, therefore, if you are storing image attributes, you should call the getProperties( ) method on the destination image after calling the processCopy( ) method.

See Appendix D for more information on processCopy( ) operators.

Pragmas

None.

Exceptions

ORDImageExceptions.DATA_NOT_LOCAL

This exception is raised if you call the processCopy( ) method and the imageBlob parameter is not initialized.

Examples

Copy an image, changing the file format, compression format, and content format in the destination image:

```
DECLARE
  dest_attrib      CLOB;
  image_data       BLOB;
  destination_data BLOB;
  the_Command      VARCHAR2(4000);
BEGIN
   SELECT img INTO image_data FROM timg WHERE N = 1;
   SELECT img, attributes INTO destination_data, dest_attrib FROM timg
      WHERE N = 2 FOR UPDATE;

   the_Command := 'fileFormat=tiff, compressionFormat=packbits, contentFormat=8bitlut';
   ORDSYS.ORDImage.processCopy(image_data, the_Command, destination_data);
   ORDSYS.ORDImage.getProperties(destination_data, dest_attrib);
   UPDATE timg SET img = destination_data, attributes=dest_attrib WHERE N = 2;
   COMMIT;
EXCEPTION
   WHEN OTHERS THEN
   RAISE;
END;
/
```

processCopy( ) for BFILEs

Format

```
processCopy(imageBfile IN OUT NOCOPY BFILE,
            command  IN VARCHAR2,
            dest     IN OUT NOCOPY BLOB);
```

Description

Copies an image stored internally or externally to another image stored internally in the source.localData attribute (of the embedded ORDSource object) and performs one or more image processing operations on the copy.

Parameters

imageBfile
The image data represented as a BFILE.

command
A list of image processing changes to make for the image in the new copy.

dest
The destination of the new image.

Usage Notes

See Table 6–1, " Image Processing Operators" and Table 6–2, " Additional Image Processing Operators for Raw Pixel and Foreign Images".

Calling this method processes the image into the destination BLOB from any source BFILE.

The processCopy( ) method changes image attributes, therefore, if you are storing image attributes, you should call the getProperties( ) method on the destination image after calling the processCopy( ) method.

See Appendix D for more information on processCopy( ) operators.

Pragmas

None.

Exceptions

ORDImageExceptions.NULL_DESTINATION

This exception is raised if you call the processCopy( ) method and the destination image is NULL.

ORDImageExceptions.NULL_LOCAL_DATA

This exception is raised when the imageBfile parameter is NULL.

Examples

Copy an image, generating a thumbnail image of, at most, 32 x 32 pixels in the destination image:

```
DECLARE
  dest_attrib      CLOB;
  image_data       BFILE := BFILENAME('IMAGEDIR','testimg.dat');
  destination_data BLOB;
  the_Command      VARCHAR2(4000);
BEGIN
   SELECT img, attributes INTO destination_data, dest_attrib FROM timg
       WHERE N = 2 FOR UPDATE;

   the_Command := 'maxScale=32 32';
   ORDSYS.ORDImage.processCopy(image_data, the_Command, destination_data);
   ORDSYS.ORDImage.getProperties(destination_data, dest_attrib);
   UPDATE timg SET img = destination_data, attributes=dest_attrib WHERE N = 2;
   COMMIT;
EXCEPTION
   WHEN OTHERS THEN
   RAISE;
END;
/
```

Static Methods Unique to the ORDVideo Object Type Relational Interface

This section presents reference information on the *inter*Media static methods unique to the ORDVideo relational interface.

The relational interface adds *inter*Media support to video data stored in BLOBs and BFILEs rather than in the ORDVideo object type. The following interface is defined in the ordvspec.sql file:

```
.
.
.
  -- Static Methods for the relational interface
  STATIC PROCEDURE export(ctx             IN OUT RAW,
                          local_data      IN BLOB,
                          source_type     IN VARCHAR2,
                          source_location IN VARCHAR2,
                          source_name     IN VARCHAR2),
  --
  STATIC PROCEDURE importFrom(ctx             IN OUT RAW,
                              local_data      IN OUT NOCOPY BLOB,
                              source_type     IN VARCHAR2,
                              source_location IN VARCHAR2,
                              source_name     IN VARCHAR2),
 --
  STATIC PROCEDURE importFrom(ctx             IN OUT RAW,
                              local_data      IN OUT NOCOPY BLOB,
                              source_type     IN VARCHAR2,
                              source_location IN VARCHAR2,
                              source_name     IN VARCHAR2,
                              format          OUT VARCHAR2,
                              mime_type       OUT VARCHAR2),
  --
  STATIC PROCEDURE getProperties(ctx            IN OUT RAW,
                                 videoBlob      IN BLOB,
                                 attributes     IN OUT NOCOPY CLOB,
                                 format         IN VARCHAR2),
  --
  STATIC PROCEDURE getProperties(ctx              IN OUT RAW,
                                 videoBlob        IN BLOB,
                                 attributes       IN OUT NOCOPY CLOB,
                                 mimeType         OUT VARCHAR2,
                                 format           IN OUT VARCHAR2,
```

```
                                width            OUT INTEGER,
                                height           OUT INTEGER,
                                frameResolution  OUT INTEGER,
                                frameRate        OUT INTEGER,
                                videoDuration    OUT INTEGER,
                                numberOfFrames   OUT INTEGER,
                                compressionType  OUT VARCHAR2,
                                numberOfColors   OUT INTEGER,
                                bitRate          OUT INTEGER),

 --
  STATIC PROCEDURE getProperties(ctx            IN OUT RAW,
                                videoBfile      IN OUT NOCOPY BFILE,
                                attributes      IN OUT NOCOPY CLOB,
                                format          IN VARCHAR2),
  --
  STATIC PROCEDURE getProperties(ctx              IN OUT RAW,
                                videoBfile        IN OUT NOCOPY BFILE,
                                attributes        IN OUT NOCOPY CLOB,
                                mimeType          OUT VARCHAR2,
                                format            IN OUT VARCHAR2,
                                width             OUT INTEGER,
                                height            OUT INTEGER,
                                frameResolution   OUT INTEGER,
                                frameRate         OUT INTEGER,
                                videoDuration     OUT INTEGER,
                                numberOfFrames    OUT INTEGER,
                                compressionType   OUT VARCHAR2,
                                numberOfColors    OUT INTEGER,
                                bitRate           OUT INTEGER),
.
.
.
```

Example Video Table Definition

The methods described in this section show examples based on a test video table TVID. Refer to the TVID table definition that follows when reading through the examples:

TVID Table Definition

```
CREATE TABLE tvid(n NUMBER,
                  vid BLOB,
                  attributes CLOB,
```

```
                    mimetype VARCHAR2(4000),
                    format VARCHAR2(31),
                    width INTEGER,
                    height INTEGER,
                    frameresolution INTEGER,
                    framerate INTEGER,
                    videoduration INTEGER,
                    numberofframes INTEGER,
                    compressiontype VARCHAR2(4000),
                    numberofcolors INTEGER,
                    bitrate INTEGER)
STORAGE (INITIAL 100K NEXT 100K PCTINCREASE 0);

INSERT INTO tvid VALUES(1, EMPTY_BLOB(), EMPTY_CLOB(), NULL, NULL,
NULL,NULL,NULL,NULL,NULL,NULL,NULL,NULL,NULL);
INSERT INTO tvid VALUES(2, EMPTY_BLOB(), EMPTY_CLOB(), NULL, NULL,
NULL,NULL,NULL,NULL,NULL,NULL,NULL,NULL,NULL);
COMMIT;
```

getProperties( ) for BLOBs

Format

```
getProperties(ctx          IN OUT RAW,
              videoBlob    IN BLOB,
              attributes   IN OUT NOCOPY CLOB,
              format       IN VARCHAR2);
```

Description

Reads the video BLOB data to get the values of the media attributes for supported formats, and then stores them in the input CLOB. This method populates the CLOB with an extensive set of format and application properties in XML form.

Parameters

ctx
The format plug-in context information.

videoBlob
The video data represented as a BLOB.

attributes
The CLOB to hold the XML attribute information generated by the getProperties( ) method. This CLOB is populated with an extensive set of format and application properties of the video BLOB data in XML form.

format
The format of the video data. If a non-NULL value is specified, then the format plug-in for this format type is invoked.

Usage Notes

None.

Pragmas

None.

Exceptions

ORDVideoExceptions.VIDEO_PLUGIN_EXCEPTION

This exception is raised if you call the getProperties( ) method and the video plug-in raises an exception.

ORDSourceExceptions.EMPTY_SOURCE

This exception is raised when the value of the source.local attribute is 1 or 0 (TRUE), but the value of the source.localData attribute is NULL.

Examples

Get the property information for known video attributes:

```
DECLARE
  vid_attrib CLOB;
  ctx RAW(64) :=NULL;
  vid_data BLOB;
  vid_format VARCHAR2(31) := NULL;
BEGIN
  SELECT vid, attributes INTO vid_data, vid_attrib FROM tvid WHERE N = 1 FOR UPDATE;
  ORDSYS.ORDVideo.getProperties(ctx, vid_data, vid_attrib, vid_format);

  DBMS_OUTPUT.put_line('Size of XML Annotations ' ||
                  TO_CHAR(DBMS_LOB.GETLENGTH(vid_attrib)));
  UPDATE tvid SET vid=vid_data, attributes=vid_attrib WHERE N=1;
  COMMIT;
EXCEPTION
  WHEN OTHERS THEN
  RAISE;
END;
/
```

getProperties( ) (all attributes) for BLOBs

Format

```
getProperties(ctx                IN OUT RAW,
              videoBlob          IN BLOB,
              attributes         IN OUT NOCOPY CLOB,
              mimeType           OUT VARCHAR2,
              format             IN OUT VARCHAR2
              width              OUT INTEGER,
              height             OUT INTEGER,
              frameResolution    OUT INTEGER,
              frameRate          OUT INTEGER,
              videoDuration      OUT INTEGER,
              numberOfFrames     OUT INTEGER,
              compressionType    OUT VARCHAR2,
              numberOfColors     OUT INTEGER,
              bitRate            OUT INTEGER);
```

Description

Reads the video BLOB data to get the values of the media attributes for supported formats, and then stores them in the input CLOB and returns them as explicit parameters. This method gets the properties for the following attributes of the video data: MIME type, format, frame size, frame resolution, frame rate, video duration, number of frames, compression type, number of colors, and bit rate. It populates the CLOB with an extensive set of format and application properties in XML form.

Parameters

ctx
The format plug-in context information.

videoBlob
The video data represented as a BLOB.

attributes
The CLOB to hold the XML attribute information generated by the getProperties( ) method. This CLOB is populated with an extensive set of format and application properties of the video BLOB data in XML form.

mimeType
The MIME type of the video data.

format
The format of the video data. If a non-NULL value is specified, then the format plug-in for this format type is invoked. If specified as NULL, the format of the video data is returned.

width
The width of the frame in pixels of the video data.

height
The height of the frame in pixels of the video data.

frameResolution
The number of pixels per inch of frames in the video data.

frameRate
The number of frames per second at which the video data was recorded.

videoDuration
The total time required to play the video data.

numberOfFrames
The total number of frames in the video data.

compressionType
The compression type of the video data.

numberOfColors
The number of colors in the video data.

bitRate
The bit rate in the video data.

Usage Notes

If a property cannot be extracted from the media source, then the respective parameter is set to NULL.

Pragmas

None.

Exceptions

ORDVideoExceptions.VIDEO_PLUGIN_EXCEPTION

This exception is raised if you call the getProperties( ) method and the video plug-in raises an exception.

ORDSourceExceptions.EMPTY_SOURCE

This exception is raised when the value of the source.local attribute is 1 or 0 (TRUE), but the value of the source.localData attribute is NULL.

Examples

Get the property information for known video attributes:

```
DECLARE
  vid_attrib        CLOB;
  ctx               RAW(64) :=NULL;
  vid_data          BLOB;
  mimeType          VARCHAR2(80);
  format            VARCHAR2(32);
  width             NUMBER;
  height            NUMBER;
  frameResolution   NUMBER;
  frameRate         NUMBER;
  videoDuration     NUMBER;
  numberOfFrames    NUMBER;
  compressionType   VARCHAR2(160);
  numberOfColors    NUMBER;
  bitRate           NUMBER;
BEGIN
  SELECT vid, attributes, mimetype, format, width, height, frameresolution, framerate,
         videoduration, numberofframes, compressiontype, numberofcolors, bitrate INTO
         vid_data, vid_attrib, mimeType, format, width, height, frameResolution,
         frameRate, videoDuration, numberOfFrames, compressionType, numberOfColors,
         bitRate FROM tvid WHERE N = 1 FOR UPDATE;

  ORDSYS.ORDVideo.getProperties(ctx, vid_data, vid_attrib, mimeType, format,
          width, height, frameResolution, frameRate,
```

```
        videoDuration, numberOfFrames, compressionType, numberOfColors, bitRate);

  DBMS_OUTPUT.put_line('Size of XML Annotations ' ||
                 TO_CHAR(DBMS_LOB.GETLENGTH(vid_attrib)));
  DBMS_OUTPUT.put_line('mimeType: ' || mimeType );
  DBMS_OUTPUT.put_line('format: ' || format );
  DBMS_OUTPUT.put_line('width: ' || width );
  DBMS_OUTPUT.put_line('height: ' || height );
  DBMS_OUTPUT.put_line('frameResolution: ' || frameResolution );
  DBMS_OUTPUT.put_line('frameRate: ' || frameRate );
  DBMS_OUTPUT.put_line('videoDuration: ' || videoDuration );
  DBMS_OUTPUT.put_line('numberOfFrames: ' || numberOfFrames );
  DBMS_OUTPUT.put_line('compressionType: ' || compressionType );
  DBMS_OUTPUT.put_line('numberOfColors: ' || numberOfColors );
  DBMS_OUTPUT.put_line('bitRate: ' || bitRate );
  UPDATE tvid SET
        vid=vid_data,
        attributes=vid_attrib,
        mimetype=mimeType,
        format=format,
        width=width,
        height=height,
        frameresolution=frameResolution,
        framerate=frameRate,
        videoduration=videoDuration,
        numberofframes=numberOfFrames,
        compressiontype=compressionType,
        numberofcolors=numberOfColors,
        bitrate=bitRate
     WHERE N=1;
  COMMIT;
EXCEPTION
  WHEN OTHERS THEN
  RAISE;
END;
/
```

getProperties( ) for BFILEs

Format

```
getProperties(ctx              IN OUT RAW,
              videoBfile       IN OUT NOCOPY BFILE,
              attributes       IN OUT NOCOPY CLOB,
              format           IN VARCHAR2);
```

Description

Reads the video BFILE data to get the values of the media attributes for supported formats, and then stores them in the input CLOB. This method populates the CLOB with an extensive set of format and application properties in XML form.

Parameters

ctx
The format plug-in context information.

videoBfile
The video data represented as a BFILE.

attributes
The CLOB to hold the XML attribute information generated by the getProperties( ) method. This CLOB is populated with an extensive set of format and application properties of the video BFILE data in XML form.

format
The format of the video data. If a non-NULL value is specified, then the format plug-in for this format type is invoked.

Usage Notes

None.

Pragmas

None.

Exceptions

ORDVideoExceptions.VIDEO_PLUGIN_EXCEPTION

This exception is raised if you call the getProperties( ) method and the video plug-in raises an exception.

ORDSourceExceptions.EMPTY_SOURCE

This exception is raised when the value of the source.local attribute is 1 or 0 (TRUE), but the value of the source.localData attribute is NULL.

Examples

Get the property information for known video attributes:

```
DECLARE
  vid_attrib CLOB;
  ctx RAW(64) :=NULL;
  vid_data BFILE := BFILENAME('VIDEODIR','testvid.dat');
  vid_format VARCHAR2(160) := NULL;
BEGIN
  DBMS_LOB.CREATETEMPORARY(vid_attrib, FALSE, DBMS_LOB.CALL);
  ORDSYS.ORDVideo.getProperties(ctx, vid_data, vid_attrib, vid_format);

  DBMS_OUTPUT.put_line('Size of XML Annotations ' ||
                  TO_CHAR(DBMS_LOB.GETLENGTH(vid_attrib)));
EXCEPTION
  WHEN OTHERS THEN
  RAISE;
END;
/
```

getProperties( ) (all attributes) for BFILEs

Format

```
getProperties(ctx                IN OUT RAW,
              videoBfile         IN OUT NOCOPY BFILE,
              attributes         IN OUT NOCOPY CLOB,
              mimeType           OUT VARCHAR2,
              format             IN OUT VARCHAR2,
              width              OUT INTEGER,
              height             OUT INTEGER,
              frameResolution    OUT INTEGER,
              frameRate          OUT INTEGER,
              videoDuration      OUT INTEGER,
              numberOfFrames     OUT INTEGER,
              compressionType    OUT VARCHAR2,
              numberOfColors     OUT INTEGER,
              bitRate            OUT INTEGER);
```

Description

Reads the video BFILE data to get the values of the media attributes for supported formats, and then stores them in the input CLOB and returns them as explicit parameters. This method gets the properties for the following attributes of the video data: MIME type, format, frame size, frame resolution, frame rate, video duration, number of frames, compression type, number of colors, and bit rate. It populates the CLOB with an extensive set of format and application properties in XML form.

Parameters

ctx
The format plug-in context information.

videoBfile
The video data represented as a BFILE.

attributes
The CLOB to hold the XML attribute information generated by the getProperties( ) method. This CLOB is populated with an extensive set of format and application properties of the video BFILE data in XML form.

mimeType
The MIME type of the video data.

format
The format of the video data. If a non-NULL value is specified, then the format plug-in for this format type is invoked. If specified as NULL, the format of the video data is returned.

width
The width of the frame in pixels of the video data.

height
The height of the frame in pixels of the video data.

frameResolution
The number of pixels per inch of frames in the video data.

frameRate
The number of frames per second at which the video data was recorded.

videoDuration
The total time required to play the video data.

numberOfFrames
The total number of frames in the video data.

compressionType
The compression type of the video data.

numberOfColors
The number of colors in the video data.

bitRate
The bit rate in the video data.

Usage Notes

If a property cannot be extracted from the media source, then the respective parameter is set to NULL.

Pragmas

None.

Exceptions

ORDVideoExceptions.VIDEO_PLUGIN_EXCEPTION

This exception is raised if you call the getProperties( ) method and the video plug-in raises an exception.

ORDSourceExceptions.EMPTY_SOURCE

This exception is raised when the value of the source.local attribute is 1 or 0 (TRUE), but the value of the source.localData attribute is NULL.

Examples

Get the property information for known video attributes:

```
DECLARE
  vid_attrib        CLOB;
  ctx               RAW(64) :=NULL;
  vid_data          BFILE := BFILENAME('VIDEODIR','testvid.dat');
  mimeType          VARCHAR2(80);
  format            VARCHAR2(32) := NULL;
  width             NUMBER;
  height            NUMBER;
  frameResolution   NUMBER;
  frameRate         NUMBER;
  videoDuration     NUMBER;
  numberOfFrames    NUMBER;
  compressionType   VARCHAR2(160);
  numberOfColors    NUMBER;
  bitRate           NUMBER;
BEGIN
  DBMS_LOB.CREATETEMPORARY(vid_attrib, FALSE, DBMS_LOB.CALL);

  ORDSYS.ORDVideo.getProperties(ctx, vid_data, vid_attrib, mimeType, format,
         width, height, frameResolution, frameRate,
         videoDuration, numberOfFrames, compressionType, numberOfColors, bitRate);

  DBMS_OUTPUT.put_line('Size of XML Annotations ' ||
                  TO_CHAR(DBMS_LOB.GETLENGTH(vid_attrib)));
```

```
  DBMS_OUTPUT.put_line('mimeType: ' || mimeType );
  DBMS_OUTPUT.put_line('format: ' || format );
  DBMS_OUTPUT.put_line('width: ' || width );
  DBMS_OUTPUT.put_line('height: ' || height );
  DBMS_OUTPUT.put_line('frameResolution: ' || frameResolution );
  DBMS_OUTPUT.put_line('frameRate: ' || frameRate );
  DBMS_OUTPUT.put_line('videoDuration: ' || videoDuration );
  DBMS_OUTPUT.put_line('numberOfFrames: ' || numberOfFrames );
  DBMS_OUTPUT.put_line('compressionType: ' || compressionType );
  DBMS_OUTPUT.put_line('numberOfColors: ' || numberOfColors );
  DBMS_OUTPUT.put_line('bitRate: ' || bitRate );
EXCEPTION
  WHEN OTHERS THEN
  RAISE;
END;
/
```

10

ORDSource

Oracle *inter*Media (*"inter*Media") contains the following information about the ORDSource object type:

This object is used only by other Oracle *inter*Media objects. The information in this chapter is included for reference only. Oracle Corporation does not recommend that you use this type.

Methods invoked at the ORDSource level that are handed off to the source plug-in for processing have ctx (RAW) as the first argument. Before calling any of these methods for the first time, the client should allocate the ctx structure, initialize it to NULL, and invoke the open( ) method. At this point, the source plug-in can initialize context for this client. When processing is complete, the client should invoke the close( ) method.

Methods invoked from a source plug-in call have the first argument as obj (ORDSource) and the second argument as ctx (RAW).

> **Note:** In the current release, none of the plug-ins provided by Oracle and not all source or format plug-ins will use the ctx argument, but if you code as previously described, your application should work with current or future source or format plug-ins.

The ORDSource object does not attempt to maintain consistency, for example, with local and upDateTime attributes. It is up to you to maintain consistency. ORDAudio, ORDDoc, ORDImage, and ORDVideo objects all maintain consistency of their included ORDSource object.

ORDSource Object Type

The ORDSource object type supports access to a variety of sources of multimedia data. It supports access to data sources locally in a BLOB within the database, externally from a BFILE on a local file system, externally from a URL on an HTTP server, or externally from a user-defined source on another server. This object type is defined as follows:

```
CREATE OR REPLACE TYPE ORDsource
AS OBJECT
(
  -- ATTRIBUTES
localData           BLOB,
srcType             VARCHAR2(4000),
srcLocation         VARCHAR2(4000),
srcName             VARCHAR2(4000),
updateTime          DATE,
local               NUMBER,
  -- METHODS
-- Methods associated with the local attribute
MEMBER PROCEDURE setLocal,
MEMBER PROCEDURE clearLocal,
MEMBER FUNCTION isLocal RETURN BOOLEAN,
PRAGMA RESTRICT_REFERENCES(isLocal, WNDS, WNPS, RNDS, RNPS),
-- Methods associated with the updateTime attribute
MEMBER FUNCTION getUpdateTime RETURN DATE,
PRAGMA RESTRICT_REFERENCES(getUpdateTime, WNDS, WNPS, RNDS, RNPS),
MEMBER PROCEDURE setUpdateTime(current_time DATE),
-- Methods associated with the source information
MEMBER PROCEDURE setSourceInformation(
                                    source_type     IN VARCHAR2,
                                    source_location IN VARCHAR2,
                                    source_name     IN VARCHAR2),
MEMBER FUNCTION getSourceInformation RETURN VARCHAR2,
PRAGMA RESTRICT_REFERENCES(getSourceInformation, WNDS, WNPS, RNDS, RNPS),

MEMBER FUNCTION getSourceType RETURN VARCHAR2,
PRAGMA RESTRICT_REFERENCES(getSourceType, WNDS, WNPS, RNDS, RNPS),

MEMBER FUNCTION getSourceLocation RETURN VARCHAR2,
PRAGMA RESTRICT_REFERENCES(getSourceLocation, WNDS, WNPS, RNDS, RNPS),

MEMBER FUNCTION getSourceName RETURN VARCHAR2,
```

```
PRAGMA RESTRICT_REFERENCES(getSourceName, WNDS, WNPS, RNDS, RNPS),

MEMBER FUNCTION getBFile RETURN BFILE,
PRAGMA RESTRICT_REFERENCES(getBFile, WNDS, WNPS, RNDS, RNPS),

-- Methods associated with source import/export operations
MEMBER PROCEDURE import(
                       ctx      IN OUT RAW,
                       mimetype OUT VARCHAR2,
                       format   OUT VARCHAR2),
MEMBER PROCEDURE importFrom(
                            ctx             IN OUT RAW,
                            mimetype        OUT VARCHAR2,
                            format          OUT VARCHAR2,
                            source_type     IN VARCHAR2,
                            source_location IN VARCHAR2,
                            source_name     IN VARCHAR2),
MEMBER PROCEDURE export(
                       ctx             IN OUT RAW,
                       source_type     IN VARCHAR2,
                       source_location IN VARCHAR2,
                       source_name     IN VARCHAR2),
-- Methods associated with source content-related operations
MEMBER FUNCTION getContentLength(ctx IN OUT RAW) RETURN INTEGER,
PRAGMA RESTRICT_REFERENCES(getContentLength, WNDS, WNPS, RNDS, RNPS),

MEMBER FUNCTION getSourceAddress(ctx IN OUT RAW,
                                 userData IN VARCHAR2)
                RETURN VARCHAR2,
PRAGMA RESTRICT_REFERENCES(getSourceAddress, WNDS, WNPS, RNDS, RNPS),

MEMBER FUNCTION getLocalContent RETURN BLOB,
PRAGMA RESTRICT_REFERENCES(geLocalContent, WNDS, WNPS, RNDS, RNPS),

MEMBER PROCEDURE getContentInTempLob(
                                     ctx      IN OUT RAW,
                                     tempLob  IN OUT NOCOPY BLOB,
                                     mimetype OUT VARCHAR2,
                                     format   OUT VARCHAR2,
                                     duration IN PLS_INTEGER := 10,
                                     cache    IN BOOLEAN := TRUE),
MEMBER PROCEDURE deleteLocalContent,

-- Methods associated with source access methods
MEMBER FUNCTION open(userArg IN RAW, ctx OUT RAW) RETURN INTEGER,
```

```
MEMBER FUNCTION close(ctx IN OUT RAW) RETURN INTEGER,
MEMBER FUNCTION trim(ctx     IN OUT RAW,
                     newlen  IN INTEGER) RETURN INTEGER,

-- Methods associated with content read/write operations
MEMBER PROCEDURE read(
                      ctx      IN OUT RAW,
                      startPos IN INTEGER,
                      numBytes IN OUT INTEGER,
                      buffer   OUT RAW),
MEMBER PROCEDURE write(
                       ctx      IN OUT RAW,
                       startPos IN INTEGER,
                       numBytes IN OUT INTEGER,
                       buffer   IN RAW),
-- Methods associated with any commands to be sent to the external source
MEMBER FUNCTION processCommand(
                               ctx     IN OUT RAW,
                               command IN VARCHAR2,
                               arglist IN VARCHAR2,
                               result  OUT RAW)
               RETURN RAW
);
```

where:

- localData: the locally stored multimedia data stored as a BLOB within the object. Up to 4 gigabytes of data can be stored as a BLOB within Oracle Database and is protected by the Oracle security and transaction environment.
- srcType: the data source type. Supported values for srcType are:

srcType Value	Description
file	A BFILE on a local file system
HTTP	An HTTP server
<name>	User-defined

Note: The srcType "file" value is a reserved word for the BFILE source plug-in provided by Oracle Corporation. To implement your own file plug-in, select a different name, for example, MYFILE.

The srcType "HTTP" value is a reserved word for the HTTP source plug-in provided by Oracle Corporation.

- srcLocation: the place where data can be found based on the srcType value. Valid srcLocation values for corresponding srcType values are:

srcType	Location Value
file	<DIR> or name of the directory object
HTTP	<SourceBase> or URL needed to find the base directory (without the string "http://")
<name>	<iden> or identifier string required to access a user-defined source

- srcName: the data object name. Valid srcName values for corresponding srcType values are:

srcType	Name Value
file	<file> or name of the file
HTTP	<Source> or name of the object
<name>	<object name> or name of the object

- updateTime: the time at which the data was last updated.
- local: a flag to determine whether or not the data is local:

  1 means the data is in the BLOB.

  0 means the data is in external sources.

  NULL, which may be a default state when you first insert an empty row, is assumed to mean data is local.

ORDSource Methods

This section presents ORDSource reference information on the ORDSource methods provided for source data manipulation, as follows:

- trim( ) on page 10-42
- write( ) on page 10-44

For more information on object types and methods, see *Oracle Database Concepts*.

clearLocal( )

Format

clearLocal( );

Description

Resets the local attribute value from 1, meaning the source of the data is stored locally in a BLOB in the database, to 0, meaning the source of the data is stored externally.

Parameters

None.

Usage Notes

This method sets the local attribute to 0, meaning the data is stored externally or outside the database.

Pragmas

None.

Exceptions

None.

Examples

None.

close( )

Format

close(ctx IN OUT RAW) RETURN INTEGER;

Description

Closes a data source.

Parameters

ctx
The source plug-in context information.

Usage Notes

Calling this method uses the ORDPLUGINS.ORDX_<srcType>_SOURCE plug-in package.

The return INTEGER is 0 (zero) for success and greater than 0 (for example, 1) for failure. The exact number and the meaning for that number is plug-in defined. For example, for the file plug-in, 1 might mean "File not found," 2 might mean "No such directory," and so forth.

Pragmas

None.

Exceptions

ORDSourceExceptions.INCOMPLETE_SOURCE_INFORMATION

This exception is raised if you call the close( ) method and the value for the srcType attribute is NULL.

ORDSourceExceptions.METHOD_NOT_SUPPORTED

This exception is raised if you call the close( ) method and this method is not supported by the source plug-in being used.

ORDSourceExceptions.SOURCE_PLUGIN_EXCEPTION

This exception is raised if you call the close( ) method and the source plug-in raises an exception.

Examples

None.

deleteLocalContent( )

Format

```
deleteLocalContent( );
```

Description

Deletes the local data from the localData attribute.

Parameters

None.

Usage Notes

This method can be called after you export the data from the local source to an external data source and you no longer need this data in the local source.

Pragmas

None.

Exceptions

None.

Examples

None.

export( )

Format

```
export(ctx             IN OUT RAW,
       source_type     IN VARCHAR2,
       source_location IN VARCHAR2,
       source_name     IN VARCHAR2);
```

Description

Copies data from the localData attribute within the database to an external data source.

Note: The export( ) method natively supports only sources of source type "file". User-defined sources may support the export( ) method.

Parameters

ctx
The source plug-in context information.

source_type
The type of the source data to be exported.

source_location
The location to which the source data is to be exported.

source_name
The name of the object to which the source data is to be exported.

Usage Notes

This method exports data out of the localData attribute to another source.

Calling this method uses the ORDPLUGINS.ORDX_<srcType>_SOURCE plug-in package.

After exporting data, the srcType, srcLocation, and srcName attributes are updated with input parameter values. After calling the export( ) method, call the clearLocal( ) method to indicate the data is stored outside the database and call the deleteLocalContent( ) method if you want to delete the content of the local data.

This method is also available for user-defined sources that can support the export( ) method.

The only server-side native support for the export method is for the srcType file.

The export( ) method for a source type of file is similar to a file copy operation in that the original data stored in the BLOB is not touched other than for reading purposes.

The export( ) method is not an exact mirror operation to the import( ) method in that the clearLocal( ) method is not automatically called to indicate the data is stored outside the database, whereas the import( ) method automatically calls the setLocal( ) method.

Call the deleteLocalContent( ) method after calling the export( ) method to delete the content from the database if you no longer intend to manage the multimedia data within the database.

The export( ) method writes only to a database directory object that the user has privilege to access. That is, you can access a directory that you have created using the SQL CREATE DIRECTORY statement, or one to which you have been granted READ access. To execute the CREATE DIRECTORY statement, you must have the CREATE ANY DIRECTORY privilege. In addition, you must use the DBMS_JAVA.GRANT_PERMISSION call to specify to which files the user and ORDSYS can write. The user must be granted the write permission so that he or she can write to the file; ORDSYS must be granted the write permission so that it can export the file on behalf of the user.

For example, the following SQL*Plus commands grant the user, MEDIAUSER, and ORDSYS the permission to write to the file named filename.dat:

```
CALL DBMS_JAVA.GRANT_PERMISSION(
        'MEDIAUSER',
        'java.io.FilePermission',
        '/audio/movies/filename.dat',
        'write');
CALL DBMS_JAVA.GRANT_PERMISSION(
        'ORDSYS',
        'java.io.FilePermission',
        '/audio/movies/filename.dat',
        'write');
```

The previous example shows how to authorize access to write to a single file. In addition, there are various wildcard path specifications that authorize write access to multiple directories and file names. For example, a path specification that ends in a slash and asterisk (/*) (where the slash is the file-separator character that is operating-system dependent) indicates all the files contained in the specified directory. A path specification that ends with a slash and a hyphen (/-) indicates all files contained in the specified directory and all its subdirectories. A path name consisting of the special token <<ALL FILES>> authorizes access to any file.

See *Oracle Database Java Developer's Guide* and the javia.io.FilePermissions class in the Java API for more information about security and performance.

Invoking this method implicitly calls the setUpdateTime( ) method.

Pragmas

None.

Exceptions

ORDSourceExceptions.INCOMPLETE_SOURCE_INFORMATION

This exception is raised if you call the export( ) method and the value of the srcType attribute is NULL.

ORDSourceExceptions.METHOD_NOT_SUPPORTED

This exception is raised if you call the export( ) method and this method is not supported by the source plug-in being used.

ORDSourceExceptions.SOURCE_PLUGIN_EXCEPTION

This exception is raised if you call the export( ) method and the source plug-in raises an exception.

Examples

None.

getBFile( )

Format

getBFile( ) RETURN BFILE;

Description

Returns a BFILE handle, if the srcType attribute value is "file".

Parameters

None.

Usage Notes

This method can be used only for a srcType of "file".

Pragmas

PRAGMA RESTRICT_REFERENCES(getBFile, WNDS, WNPS, RNDS, RNPS)

Exceptions

ORDSourceExceptions.INCOMPLETE_SOURCE_INFORMATION

This exception is raised if you call the getBFile( ) method and the value of the srcType attribute is NULL.

ORDSourceExceptions.INVALID_SOURCE_TYPE

This exception is raised if you call the getBFile( ) method and the value of the srcType attribute is other than "file".

Examples

None.

getContentInTempLob( )

Format

```
getContentInTempLob(ctx          IN OUT RAW,
                    tempLob   IN OUT NOCOPY BLOB,
                    mimeType OUT VARCHAR2,
                    format      OUT VARCHAR2,
                    duration    IN PLS_INTEGER := 10,
                    cache       IN BOOLEAN := TRUE);
```

Description

Transfers data from the current data source into a temporary LOB, which will be allocated and initialized as a part of this call.

Parameters

ctx
The source plug-in context information.

tempLob
An uninitialized BLOB locator, which will be allocated in this call.

mimeType
An output parameter to receive the MIME type of the data, for example, 'audio/basic'.

format
An output parameter to receive the format of the data, for example, 'AUFF'.

duration
The life of the temporary LOB to be allocated. The life of the temporary LOB can be for the duration of the call, the transaction, or for the session. The default is DBMS_LOB.SESSION. Valid values for each duration state are as follow:

DBMS_LOB.CALL

DBMS_LOB.TRANSACTION

DBMS_LOB.SESSION

cache
Whether or not you want to keep the data cached. The value is either TRUE or FALSE. The default is TRUE.

Usage Notes

None.

Pragmas

None.

Exceptions

NO_DATA_FOUND

This exception is raised if you call the getContentInLob( ) method when working with temporary LOBs for looping read operations that reach the end of the LOB, and there are no more bytes to be read from the LOB. (There is no ORD<*object-type*>Exceptions prefix to this exception because it is a predefined PL/SQL exception.)

ORDSourceExceptions.SOURCE_PLUGIN_EXCEPTION

This exception is raised if you call the getContentInLob( ) method and the source plug-in raises an exception.

Examples

None.

getContentLength( )

Format

getContentLength(ctx IN OUT RAW) RETURN INTEGER;

Description

Returns the length of the data content stored in the source. For a file source and for data in the localData attribute, the length is returned as a number of bytes. The unit type of the returned value is defined by the plug-in that implements this method.

Parameters

ctx
The source plug-in context information.

Usage Notes

Calling this method uses the ORDPLUGINS.ORDX_<srcType>_SOURCE plug-in package.

Pragmas

PRAGMA RESTRICT_REFERENCES(getContentLength, WNDS, WNPS, RNDS, RNPS)

Exceptions

ORDSourceExceptions.INCOMPLETE_SOURCE_INFORMATION

This exception is raised if you call the getContentLength( ) method and the value of the srcType attribute is NULL.

ORDSourceExceptions.SOURCE_PLUGIN_EXCEPTION

This exception is raised if you call the getContentLength( ) method and the source plug-in raises an exception.

Examples

None.

getLocalContent( )

Format

getLocalContent( ) RETURN BLOB;

Description

Returns the content or BLOB handle of the localData attribute.

Parameters

None.

Usage Notes

None.

Pragmas

PRAGMA RESTRICT_REFERENCES(getLocalContent, WNDS, WNPS, RNDS, RNPS)

Exceptions

None.

Examples

None.

getSourceAddress( )

Format

```
getSourceAddress(ctx IN OUT RAW,
                 userData IN VARCHAR2)  RETURN VARCHAR2;
```

Description

Returns the source address for data located in an external data source. This method is implemented only for user-defined sources.

Parameters

ctx
The source plug-in context information.

userData
User input needed by some sources to obtain the desired source address.

Usage Notes

Use this method to return the address of an external data source when the source needs to format this information in some unique way. For example, call the getSourceAddress( ) method to obtain the address for RealNetworks server sources or URLs containing data sources located on Oracle Application Server.

Calling this method uses the ORDPLUGINS.ORDX_<srcType>_SOURCE plug-in package.

Pragmas

PRAGMA RESTRICT_REFERENCES(getSourceAddress, WNDS,
WNPS, RNDS, RNPS)

Exceptions

ORDSourceExceptions.INCOMPLETE_SOURCE_INFORMATION

This exception is raised if you call the getSourceAddress( ) method and the value of the srcType attribute is NULL.

ORDSourceExceptions.SOURCE_PLUGIN_EXCEPTION

This exception is raised if you call the getSourceAddress( ) method and the source plug-in raises an exception.

Examples

None.

getSourceInformation( )

Format

```
getSourceInformation( ) RETURN VARCHAR2;
```

Description

Returns a URL formatted string containing complete information about the external data source.

Parameters

None.

Usage Notes

This method returns a VARCHAR2 string formatted as: <srcType>://<srcLocation>/<srcName>, where srcType, srcLocation, and srcName are the ORDSource attribute values.

Pragmas

PRAGMA RESTRICT_REFERENCES(getSourceInformation, WNDS, WNPS, RNDS, RNPS)

Exceptions

None.

Examples

None.

getSourceLocation( )

Format

```
getSourceLocation( ) RETURN VARCHAR2;
```

Description

Returns the external data source location.

Parameters

None.

Usage Notes

This method returns the current value of the srcLocation attribute, for example BFILEDIR.

Pragmas

PRAGMA RESTRICT_REFERENCES(getSourceLocation, WNDS, WNPS, RNDS, RNPS)

Exceptions

ORDSourceExceptions.INCOMPLETE_SOURCE_INFORMATION

This exception is raised if you call the getSourceLocation( ) method and the value of the srcType attribute is NULL.

ORDSourceExceptions.INCOMPLETE_SOURCE_LOCATION

This exception is raised if you call the getSourceLocation( ) method and the value of the srcLocation attribute is NULL.

Examples

None.

getSourceName( )

Format

```
getSourceName( ) RETURN VARCHAR2;
```

Description

Returns the external data source name.

Parameters

None.

Usage Notes

This method returns the current value of the srcName attribute, for example testaud.dat.

Pragmas

PRAGMA RESTRICT_REFERENCES(getSourceName, WNDS, WNPS, RNDS, RNPS)

Exceptions

ORDSourceExceptions.INCOMPLETE_SOURCE_INFORMATION

This exception is raised if you call the getSourceName( ) method and the value of the srcType attribute is NULL.

ORDSourceExceptions.INCOMPLETE_SOURCE_NAME

This exception is raised if you call the getSourceName( ) method and the value of the srcName attribute is NULL.

Examples

None.

getSourceType( )

Format

getSourceType( ) RETURN VARCHAR2;

Description

Returns the external data source type.

Parameters

None.

Usage Notes

This method returns the current value of the srcType attribute, for example "file".

Pragmas

PRAGMA RESTRICT_REFERENCES(getSourceType, WNDS, WNPS, RNDS, RNPS)

Exceptions

None.

Examples

None.

getUpdateTime( )

Format

```
getUpdateTime( ) RETURN DATE;
```

Description

Returns the timestamp of when the object was last changed, or what the user explicitly set by calling the setUpdateTime( ) method. (This method returns the value of the updateTime attribute.)

Parameters

None.

Usage Notes

None.

Pragmas

PRAGMA RESTRICT_REFERENCES(getUpdateTime, WNDS, WNPS, RNDS, RNPS)

Exceptions

None.

Examples

None.

import( )

Format

```
import(ctx          IN OUT RAW,
       mimeType OUT VARCHAR2,
       format     OUT VARCHAR2);
```

Description

Transfers data from an external data source (specified by first calling setSourceInformation( )) to the localData attribute within the database.

Parameters

ctx
The source plug-in context information.This information is passed along uninterpreted to the source plug-in handling the import( ) call.

mimeType
The output parameter to receive the MIME type of the data, if any, for example, audio/basic.

format
The output parameter to receive the format of the data, if any, for example, AUFF.

Usage Notes

Call setSourceInformation( ) to set the srcType, srcLocation, and srcName attribute values to describe where the data source is located prior to calling the import( ) method.

Calling this method uses the ORDPLUGINS.ORDX_<srcType>_SOURCE plug-in package.

This method uses the PL/SQL UTL_HTTP package to import media data from an HTTP data source. You can use environment variables to specify the proxy behavior of the UTL_HTTP package. For example, on UNIX, setting the environment variable http_proxy to a URL specifies that the UTL_HTTP package will use that URL as the proxy server for HTTP requests. Setting the no_proxy environment variable to a domain name specifies that the HTTP proxy server will not be used for URLs in the specified domain.

See *PL/SQL Packages and Types Reference* for more information about the UTL_HTTP PL/SQL package.

Pragmas

None.

Exceptions

ORDSourceExceptions.INCOMPLETE_SOURCE_INFORMATION

This exception is raised if you call the import( ) method and the value of the srcType attribute is NULL.

ORDSourceExceptions.NULL_SOURCE

This exception is raised if you call the import( ) method and the value of the localData attribute is NULL.

ORDSourceExceptions.METHOD_NOT_SUPPORTED

This exception is raised if you call the import( ) method and this method is not supported by the source plug-in being used.

ORDSourceExceptions.SOURCE_PLUGIN_EXCEPTION

This exception is raised if you call the import( ) method and the source plug-in raises an exception.

Examples

None.

importFrom( )

Format

```
importFrom(ctx             IN OUT RAW,
           mimeType        OUT VARCHAR2,
           format          OUT VARCHAR2
           source_type     IN VARCHAR2,
           source_location IN VARCHAR2,
           source_name     IN VARCHAR2);
```

Description

Transfers data from the specified external data source (type, location, name) to a the localData attribute within the database, and resets the source attributes and the timestamp.

Parameters

ctx
The source plug-in context information. This information is passed along uninterpreted to the source plug-in handling the importFrom( ) call.

mimeType
The output parameter to receive the MIME type of the data, if any, for example, audio/basic.

format
The output parameter to receive the format of the data, if any, for example, AUFF.

source_type
The type of the source data to be imported. This also sets the srcType attribute.

source_location
The location from which the source data is to be imported. This also sets the srcLocation attribute.

source_name
The name of the source data to be imported. This also sets the srcName attribute.

Usage Notes

This method describes where the data source is located by specifying values for the type, location, and name parameters, which set the srcType, srcLocation, and srcName attribute values, respectively, after the importFrom( ) operation succeeds.

This method is a combination of a setSourceInformation( ) call followed by an import( ) call.

Calling this method uses the ORDPLUGINS.ORDX_<srcType>_SOURCE plug-in package.

This method uses the PL/SQL UTL_HTTP package to import media data from an HTTP data source. You can use environment variables to specify the proxy behavior of the UTL_HTTP package. For example, on UNIX, setting the environment variable http_proxy to a URL specifies that the UTL_HTTP package will use that URL as the proxy server for HTTP requests. Setting the no_proxy environment variable to a domain name specifies that the HTTP proxy server will not be used for URLs in the specified domain.

See *PL/SQL Packages and Types Reference* for more information about the UTL_HTTP PL/SQL package.

Pragmas

None.

Exceptions

ORDSourceExceptions.NULL_SOURCE

This exception is raised if you call the importFrom( ) method and the value of the localData attribute is NULL.

ORDSourceExceptions.METHOD_NOT_SUPPORTED

This exception is raised if you call the importFrom( ) method and this method is not supported by the source plug-in being used.

ORDSourceExceptions.SOURCE_PLUGIN_EXCEPTION

This exception is raised if you call the importFrom( ) method and the source plug-in raises an exception.

Examples

None.

isLocal( )

Format

isLocal( ) RETURN BOOLEAN;

Description

Returns TRUE if the data is stored as a BLOB locally in the localData attribute or FALSE if the data is stored externally.

Parameters

None.

Usage Notes

If the local attribute is set to 1 or NULL, this method returns TRUE, otherwise this method returns FALSE.

Pragmas

PRAGMA RESTRICT_REFERENCES(isLocal, WNDS, WNPS, RNDS, RNPS)

Exceptions

None.

Examples

None.

open( )

Format

open(userArg IN RAW, ctx OUT RAW) RETURN INTEGER;

Description

Opens a data source. It is recommended that this method be called before invoking any other methods that accept the ctx parameter.

Parameters

userArg
The user-defined input parameter.

ctx
The source plug-in context information.

Usage Notes

Calling this method uses the ORDPLUGINS.ORDX_<srcType>_SOURCE plug-in package.

The return INTEGER is 0 (zero) for success and greater than 0 (for example, 1) for failure. The exact number and the meaning for that number is plug-in defined. For example, for the file plug-in, 1 might mean "File not found," 2 might mean "No such directory," and so forth.

Pragmas

None.

Exceptions

ORDSourceExceptions.INCOMPLETE_SOURCE_INFORMATION

This exception is raised if you call the open( ) method and the value for the srcType attribute is NULL.

ORDSourceExceptions.METHOD_NOT_SUPPORTED

This exception is raised if you call the open( ) method and this method is not supported by the source plug-in being used.

ORDSourceExceptions.SOURCE_PLUGIN_EXCEPTION

This exception is raised if you call the open( ) method and the source plug-in raises an exception.

Examples

None.

processCommand( )

Format

```
processCommand(ctx          IN OUT RAW,
               command IN VARCHAR2,
               arglist      IN VARCHAR2,
               result       OUT RAW)
RETURN RAW;
```

Description

Lets you send commands and related arguments to the source plug-in. This method is supported only for user-defined sources.

Parameters

ctx
The source plug-in context information.

command
Any command recognized by the source plug-in.

arglist
The arguments for the command.

result
The result of calling this method returned by the plug-in.

Usage Notes

Use this method to send any commands and their respective arguments to the plug-in. Commands are not interpreted; they are taken and passed through to be processed.

Calling this method uses the ORDPLUGINS.ORDX_<srcType>_SOURCE plug-in package.

Pragmas

None.

Exceptions

ORDSourceExceptions.INCOMPLETE_SOURCE_INFORMATION

This exception is raised if you call the processCommand( ) method and the value of the srcType attribute is NULL.

ORDSourceExceptions.METHOD_NOT_SUPPORTED

This exception is raised if you call the processCommand( ) method and this method is not supported by the source plug-in being used.

ORDSourceExceptions.SOURCE_PLUGIN_EXCEPTION

This exception is raised if you call the processCommand( ) method and the source plug-in raises an exception.

Examples

None.

read( )

Format

```
read(ctx       IN OUT RAW,
     startPos  IN INTEGER,
     numBytes IN OUT INTEGER,
     buffer    OUT RAW);
```

Description

Lets you read a buffer of numBytes from a source beginning at a start position (startPos).

Parameters

ctx
The source plug-in context information.

startPos
The start position in the data source.

numBytes
The number of bytes to be read from the data source.

buffer
The buffer to where the data will be read.

Usage Notes

This method is not supported for HTTP sources.

To successfully read HTTP source types, the entire URL source must be requested to be read. If you want to implement a read method for an HTTP source type, you must provide your own implementation for this method in the modified source plug-in for the HTTP source type.

Calling this method uses the ORDPLUGINS.ORDX_<srcType>_SOURCE plug-in package.

Pragmas

None.

Exceptions

ORDSourceExceptions.NULL_SOURCE

This exception is raised if you call the read( ) method and the value of the local attribute is 1 or NULL, but the value of the localData attribute is NULL.

ORDSourceExceptions.INCOMPLETE_SOURCE_INFORMATION

This exception is raised if you call the read( ) method and the value of the srcType attribute is NULL.

ORDSourceExceptions.METHOD_NOT_SUPPORTED

This exception is raised if you call the read( ) method and this method is not supported by the source plug-in being used.

ORDSourceExceptions.SOURCE_PLUGIN_EXCEPTION

This exception is raised if you call the read( ) method and the source plug-in raises an exception.

Examples

None.

setLocal( )

Format

```
setLocal( );
```

Description

Sets the local attribute to indicate that the BLOB data is stored in the localData attribute within the database.

Parameters

None.

Usage Notes

This method sets the local attribute to 1, meaning the data is stored locally in the localData attribute.

Pragmas

None.

Exceptions

None.

Examples

None.

setSourceInformation( )

Format

```
setSourceInformation(source_type     IN VARCHAR2,
                     source_location IN VARCHAR2,
                     source_name     IN VARCHAR2);
```

Description

Sets the provided subcomponent information for the srcType, srcLocation, and srcName attributes that describes the external data source.

Parameters

source_type
The type of the external source data. See the "ORDSource Object Type" on page 10-2 for more information.

source_location
The location of the external source data. See the "ORDSource Object Type" on page 10-2 for more information.

source_name
The name of the external source data. See the "ORDSource Object Type" on page 10-2 for more information.

Usage Notes

Before you call the import( ) method, you must call the setSourceInformation( ) method to set the srcType, srcLocation, and srcName attribute information to describe where the data source is located. If you call the importFrom( ) or the export( ) method, then these attributes are set after the importFrom( ) or export( ) call succeeds.

You must ensure that the directory indicated by the source_location parameter exists or is created before you use this method.

Pragmas

None.

Exceptions

ORDSourceExceptions.INCOMPLETE_SOURCE_INFORMATION

This exception is raised if you call the setSourceInformation( ) method and the value for the source_type parameter is NULL.

Examples

None.

setUpdateTime( )

Format

setUpdateTime(current_time DATE);

Description

Sets the value of the updateTime attribute to the time you specify.

Parameters

current_time
The update time.

Usage Notes

If current_time is NULL, updateTime is set to SYSDATE (the current time).

Pragmas

None.

Exceptions

None.

Examples

None.

trim( )

Format

```
trim(ctx IN OUT RAW,
     newlen IN INTEGER) RETURN INTEGER;
```

Description

Trims a data source.

Parameters

ctx
The source plug-in context information.

newlen
The trimmed new length.

Usage Notes

Calling this method uses the ORDPLUGINS.ORDX_<srcType>_SOURCE plug-in package.

The return INTEGER is 0 (zero) for success and greater than 0 (for example, 1) for failure. The exact number and the meaning for that number is plug-in defined. For example, for the file plug-in, 1 might mean "File not found," 2 might mean "No such directory," and so forth.

Pragmas

None.

Exceptions

ORDSourceExceptions.INCOMPLETE_SOURCE_INFORMATION

This exception is raised if you call the trim( ) method and the value for the srcType attribute is NULL.

ORDSourceExceptions.METHOD_NOT_SUPPORTED

This exception is raised if you call the trim( ) method and this method is not supported by the source plug-in being used.

ORDSourceExceptions.SOURCE_PLUGIN_EXCEPTION

This exception is raised if you call the trim( ) method and the source plug-in raises an exception.

Examples

None.

write( )

Format

```
write(ctx       IN OUT RAW,
      startPos  IN INTEGER,
      numBytes IN OUT INTEGER,
      buffer    IN RAW);
```

Description

Lets you write a buffer of numBytes to a source beginning at a start position (startPos).

Parameters

ctx
The source plug-in context information.

startPos
The start position in the source to where the buffer should be copied.

numBytes
The number of bytes to be written to the source.

buffer
The buffer of data to be written.

Usage Notes

This method assumes that the source lets you write numBytes at a random byte location. For example, the file and HTTP source types cannot be written to and do not support this method.

Calling this method uses the ORDPLUGINS.ORDX_<srcType>_SOURCE plug-in package.

Pragmas

None.

Exceptions

ORDSourceExceptions.NULL_SOURCE

This exception is raised if you call the read( ) method and the value of the local attribute is 1 or NULL, but the value of the localData attribute is NULL.

ORDSourceExceptions.INCOMPLETE_SOURCE_INFORMATION

This exception is raised if you call the write( ) method and the value of the srcType attribute is NULL.

ORDSourceExceptions.METHOD_NOT_SUPPORTED

This exception is raised if you call the write( ) method and this method is not supported by the source plug-in being used.

ORDSourceExceptions.SOURCE_PLUGIN_EXCEPTION

This exception is raised if you call the write( ) method and the source plug-in raises an exception.

Examples

None.

A

Audio File and Compression Formats

The following sections describe the audio file and compression formats and other audio features supported by Oracle *inter*Media (*"inter*Media").

Find the data format you are interested in, and then determine the supported formats. For example, Section A.1 shows that *inter*Media supports AIFF format for single channel, stereo, 8-bit and 16-bit samples, linear PCM encoding, and uncompressed format.

A.1 Supported AIFF Data Formats

The supported AIFF format ID, file format, file extension, MIME type, audio features, compression format, and encoding/compression type are the following:

- Format ID: AIFF
- File Format: AIFF
- File extension: .aff
- MIME type: audio/x-aiff
- Audio features:
  - Single channel
  - Stereo
  - 8-bit samples
  - 16-bit samples
  - Linear PCM encoding
- Compression format: Standard AIFF Uncompressed

- Encoding/Compression Type: TWOS

A.2 Supported AIFF-C Data Formats

The supported AIFF-C format ID, file format, file extension, MIME type, and audio features are the following:

- Format ID: AIFC
- File format: AIFC
- File extension: .aft
- MIME type: audio/x-aiff
- Audio features:
  - Single channel
  - Stereo
  - 8-bit samples
  - 16-bit samples

Table A–1 lists the supported AIFF-C data compression format names and encoding/compression types.

Table A–1 Supported AIFF-C Data Compression Formats and Types

Compression Formats	Encoding/Compression Types
Not compressed	Uncompressed (TWOS)
ACE 2-to-1	ACE2
ACE 8-to-3	ACE8
MACE 3-to-1	MAC3
MACE 6-to-1	MAC6

A.3 Supported AU Data Formats

The supported AU format ID, file format, file extension, MIME type, and audio features are the following:

- Format ID: AUFF
- File format: AUFF

- File extension: .au
- MIME type: audio/basic
- Audio features:
  - Single channel
  - Stereo
  - 8-bit samples
  - 16-bit samples
  - mu-law encoding
  - Linear PCM encoding

Table A–2 lists the supported AU data compression format names and encoding/compression types.

Table A–2 AU Data Compression Formats and Types

Compression Format	Encoding/Compression Types
Unspecified format	UNSPECIFIED
8-bit mu-law samples	MULAW
8-bit linear samples	LINEAR
16-bit linear samples	LINEAR
24-bit linear samples	LINEAR
32-bit linear samples	LINEAR
Floating-point samples	FLOAT
Double-precision float samples	DOUBLE
Fragmented sample data	FRAGMENTED
Nested format	NESTED
DSP program	DSP_CORE
8-bit fixed-point samples	DSP_DATA
16-bit fixed-point samples	DSP_DATA
24-bit fixed-point samples	DSP_DATA
32-bit fixed-point samples	DSP_DATA

Table A–2 (Cont.) AU Data Compression Formats and Types

Compression Format	Encoding/Compression Types
Unknown AU format	UNKNOWN
Nonaudio display data	DISPLAY
Squelch format	MULAW_SQUELCH
16-bit linear with emphasis	EMPHASIZED
16-bit linear with compression	COMPRESSED
16-bit linear with emphasis and compression	COMPRESSED_EMPHASIZED
Music Kit DSP commands	DSP_COMMANDS
DSP commands samples	DSP_COMMANDS_SAMPLES
ADPCM G721	ADPCM_G721
ADPCM G722	ADPCM_G722
ADPCM G723_3	ADPCM_G723_3
ADPCM G723_5	ADPCM_G723_5
8-bit a-law samples	ALAW

A.4 Supported Audio MPEG Data Formats

The supported audio MPEG formats are MPEG1, MPEG2, and MPEG4, as described in the following sections.

A.4.1 Supported MPEG1 and MPEG2 Data Formats

The MPEG1 and MPEG2 format ID, file format, file extension, MIME type, and audio features are the following:

- Format ID: MPEG
- File format: MPGA
- File extension: .mpg
- MIME type: audio/mpeg
- Audio Features:
  - Layer I

- Layer II
- Layer III

Table A–3 lists the supported audio MPEG1 and MPEG2 data compression format names and encoding/compression types.

Table A–3 Audio MPEG1 and MPEG2 Compression Formats and Types

Compression Formats	Encoding/Compression Types
MPEG Audio, Layer I	LAYER1
MPEG Audio, Layer II	LAYER2
MPEG Audio, Layer III	LAYER3

A.4.2 Supported MPEG4 Data Formats

The MPEG4 format ID, file format, file extension, and MIME type features are the following:

- Format ID: MP4
- File format: MP4
- File extension: .mp4
- MIME type: application/mpeg4

A.5 Supported RealNetworks Real Audio Data Format

The supported RealNetworks Real Audio format ID, file format, file extensions, MIME type, and audio features are the following:

- Format ID: RMFF
- File format: RMFF
- File extensions: .ra, .rm, and .ram
- MIME type: audio/x-pn-realaudio
- Audio features: one or more streams with different bit rates

A.6 Supported WAV Data Formats

The supported WAV format ID, file format, file extension, MIME type, and audio features are the following:

- Format ID: WAVE
- File format: WAVE
- File extension: .wav
- MIME type: audio/x-wav
- Audio features:
  - Single channel
  - Stereo
  - 8-bit samples
  - 16-bit samples
  - Linear PCM encoding

Table A–4 lists the supported WAV data compression format names and encoding/compression types.

Table A–4 WAV Data Compression Formats and Types

Compression Formats	Encoding/Compression Types
Unknown Wave Format	UNKNOWN
Microsoft PCM Wave Format	MS_PCM
Microsoft ADPCM Wave Format	MS_ADPCM
IBM CVSD Wave Format	IBM_CVSD
Microsoft aLaw Wave Format	ALAW
Microsoft mu-Law Wave Format	MULAW
OKI ADPCM Wave Format	OKI_ADPCM
Intel DVI/IMA ADPCM Wave Format	DVI_ADPCM
VideoLogic Media Space ADPCM Wave Format	MEDIASPACE_ADPCM
Sierra Semiconductor ADPCM Wave Format	SIERRA_ADPCM
Antex Electronics G723 ADPCM Wave Format	ANTEX_G723_ADPCM

Table A–4 (Cont.) WAV Data Compression Formats and Types

Compression Formats	Encoding/Compression Types
DSP Solutions DIGISTD Wave Format	DIGISTD
DSP Solutions DIGIFIX Wave Format	DIGIFIX
Dialogic OKI ADPCM Wave Format	DIALOGIC_OKI_ADPCM
Yamaha ADPCM Wave Format	YAMAHA_ADPCM
Speech Compression Sonarc Wave Format	SONARC
DSP Group TrueSpeech Wave Format	DSPGROUP_TRUESPEECH
Echo Speech Wave Format	ECHOSC1
Audiofile AF36 Wave Format	AUDIOFILE_AF36
Audio Processing Technology Wave Format	APTX
Audiofile AF10 Wave Format	AUDIOFILE_AF10
Dolby AC-2 Wave Format	DOLBY_AC2
Microsoft GSM 610 Wave Format	MS_GSM610
Antex Electronics ADPCME Wave Format	ANTEX_ADPCME
Control Resources VQLPC Wave Format	CONTROL_RES_VQLPC
DSP Solutions DIGIREAL Wave Format	DIGIREAL
DSP Solutions DIGIADPCM Wave Format	DIGIADPCM
Control Resources CR10 Wave Format	CONTROL_RES_CR10
Natural Microsystems NMS VBXADPCM Wave Format	NMS_VBXADPCM
Crystal Semiconductor IMA ADPCM Wave Format	CS_IMAADPCM
Antex Electronics G721 ADPCM Wave Format	ANTEX_G721_ADPCM
MPEG-1 Audio Wave Format	MPEG
Creative Labs ADPCM Wave Format	CREATIVE_ADPCM
Creative Labs FastSpeech8 Wave Format	CREATIVE_FASTSPEECH8
Creative Labs FastSpeech10 Wave Format	CREATIVE_FASTSPEECH10
Fujitsu FM Towns Wave Format	FM_TOWNS_SND
Olivetti GSM Wave Format	OLIGSM

Table A–4 (Cont.) WAV Data Compression Formats and Types

Compression Formats	Encoding/Compression Types
Olivetti ADPCM Wave Format	OLIADPCM
Olivetti CELP Wave Format	OLICELP
Olivetti SBC Wave Format	OLISBC
Olivetti OPR Wave Format	OLIOPR

B

Image File and Compression Formats

Descriptions of each of the Oracle *inter*Media(*"inter*Media") supported image file formats and image compression formats are presented in Section B.1 and Section B.2; the following summary tables are presented in Section B.3:

- Table B–1, " I/O Support for Image File Content Format Characteristics"
- Table B–2, " I/O Support for Image File Compression Formats"
- Table B–3, " I/O Support for Image File Formats Other Than Content and Compression"

See Appendix D for information on image formatting operators.

B.1 Image File Formats

Image file formats are listed alphabetically.

BMPF

extension: .bmp

mime: image/bmp

BMPF is the Microsoft Windows bitmap format and is based on the internal data structures used by Windows to store bitmap data in memory. This format is used extensively by Microsoft Windows, and a variant of this format is used by the IBM OS/2 operating system. Because this format is supported directly by Windows, its use is very popular in that environment and has spread to other systems.

BMPF is a very flexible image format in that it can store a wide variety of image data types, but it does not offer powerful compression. The only compression available is a run-length encoding variant that is supported only by certain content

formats. It is worth noting that BMPF is unusual in that the ordinary scanline order for this format is bottom-up, which Oracle *inter*Media calls INVERSE.

CALS

extension: .cal

mime: image/x-ora-cals

CALS is an image format for document interchange developed by the Computer-Aided Acquisition and Logistics Support office of the United States government. There are actually two variants of the CALS image format; *inter*Media supports CALS Type I. Because the CALS format is monochrome-only, it is primarily useful for storing simple documents, scanned or otherwise.

Foreign Images

Foreign images are images for which *inter*Media does not provide native recognition and support, but that can sometimes be read if the image data complies with the rules outlined in the Foreign Image Support section of the Raw Pixel appendix (see Section E.10).

FPIX

extension: .fpx

mime: image/x-fpx

FPIX, or FlashPix, is a format developed by Kodak, Microsoft Corporation, Hewlett-Packard Company, and Live Picture, Inc., for storing digital photography. FlashPix images are composed of a series of different resolutions of the same image, and each resolution is composed of individual tiles. These tiles can be uncompressed or compressed using JPEG. The multi-resolution capability of FlashPix images is intended to promote easy use in a wide variety of applications by allowing low resolution versions of the image to be used where high resolution versions are not necessary (such as browsing, viewing on screen), while high resolution versions are available when needed (printing or zooming in on an image detail).

Oracle *inter*Media includes a simple FlashPix decoder that always selects the largest resolution plane in a FlashPix image. Lower resolutions are not accessible. Oracle *inter*Media does not write FlashPix images.

GIFF

extension: .gif

mime: image/gif

GIFF is the *inter*Media name for the Graphics Interchange Format (GIF), which was developed by CompuServe to transfer images between users in their early network system. Because GIF (pronounced "jif") is an early format and was developed for use on limited hardware, it does not support content formats that store more than 8 bits per pixel. This makes the format less suitable for storing photographic or photo-realistic images than deeper formats such as PNG or JFIF, but it is a good choice for other applications. There are two specific variants of the GIF format, called 87a and 89a; *inter*Media reads both variants but writes the 87a variant.

Despite its pixel depth limitations, the GIF format remains a powerful and flexible image format, and includes support for limited transparency effects and simple animations by encoding a series of image frames and frame transition effects. Oracle *inter*Media can read GIF images that include these options but only the first frame of an animated GIF image is made available, and there is no support for writing animated GIF images.

All GIF images are compressed using a GIF-specific LZW compression scheme, which *inter*Media calls GIFLZW.

JFIF

extension: .jpg

mime: image/jpeg

JFIF is the JPEG File Interchange Format, developed by C-Cube Microsystems for storing JPEG encoded images. The JFIF format is actually just a JPEG data stream with an identifying header and a few enforced conventions. As such, it provides minimal support for anything but the actual image data. By definition, all JFIF files are JPEG compressed, making them less appropriate for some applications, as explained in the description of the JPEG compression format in Image Compression Formats.

Oracle *inter*Media identifies several distinct image formats as JFIF, including actual JFIF files, non-JFIF pure JPEG data streams, and EXIF files. The last is a JFIF variant produced by digital cameras.

PBMF, PGMF, PPMF, and PNMF

extension: .pbm, .pgm, .ppm, .pnm

mime: image/x-portable-bitmap, image/x-portable-graymap, image/x-portable-pixmap, image/x-portable-anymap

These are a family of file formats derived from Jef Poskanzer's Portable Bitmap Utilities suite. These file formats are Portable Bitmap (PBM), Portable Graymap (PGM), Portable Pixmap (PPM) and Portable Anymap (PNM). Because of their wide support and the free availability of software to handle these formats, these file formats are frequently used for uncompressed image interchange.

PBM files are monochrome only (the term "bitmap" being used in the sense of a map of bits, that is, each pixel is either 0 or 1). PGM files are grayscale only, while PPM files are full color pixel maps.

PNM does not refer to a distinct file format, but instead refers to any of the other three types (PBM, PGM, or PPM). Images written using the file format designation PNMF will be written as the most appropriate variant depending on the input data content format.

These formats do not include data compression, but have two encoding formats: ASCII or RAW.

PCXF

extension: .pcx

mime: image/pcx

PCX, or PCXF in Oracle *inter*Media notation, is an early and widely used image file format developed for ZSoft's PC Paintbrush, and later used in derivatives of that program. Despite its ancestry, it provides support for many pixel depths, from monochrome to 24-bit color. It supports a fast compression scheme designated PCXRLE by Oracle *inter*Media. Oracle *inter*Media reads but does not write PCX images.

PICT

extension: .pct

mime: image/pict

The Macintosh PICT format was developed by Apple Computer, Inc., as part of the QuickDraw toolkit built into the Macintosh ROM. It provides the ability to "record" and "playback" QuickDraw sequences, including both vector and raster graphics painting. Oracle *inter*Media supports only the raster elements of PICT files. Both Packbits and JPEG compressed PICT images are supported.

PNGF

extension: .png

mime: image/png

PNGF is the *inter*Media designation for the Portable Network Graphics (PNG) format (pronounced "ping"). PNG was developed by the PNG Development Group as a legally unencumbered and more capable replacement for some uses of the GIF and TIFF file formats. PNG includes support for deep images (up to 16 bits per sample and up to 4 samples per pixel), full alpha support, rich metadata storage including metadata compression, built-in error and gamma correction, a powerful and free compression algorithm called DEFLATE, and much more. The main feature found in GIF that is absent in PNG is the ability to store animations.

PNG support for a broad variety of pixel depths (1 bit to 16 bits per sample) makes it suitable for a very wide variety of applications, spanning the separate domains previously filled by GIF and JPEG, and being very similar to the uses of the powerful TIFF format. Because the DEFLATE compression scheme is lossless, PNG is a good choice for storing deep images that must be edited often.

All PNG images are compressed using the DEFLATE scheme.

RPIX

extension: .rpx

mime: image/x-ora-rpix

RPIX, or Raw Pixel, is a format developed by Oracle Corporation for storing simple raw pixel data without compression, and using a simple well-described header structure. It was designed to be used by applications whose native image format is not supported by *inter*Media but for which an external translation might be available. It flexibly supports N-banded image data (8 bits per sample) where N is less than 256 bands, and can handle data that is encoded in a variety of channel orders (such as RGB, BGR, BRG, and so forth), a variety of pixel orders (left-to-right and right-to-left), a variety of scanline orders (top-down or bottom-up) and a variety of band orders (band interleaved by pixel, by scanline, and by plane). The flexibility of the format includes a data offset capability, which can allow an RPIX header to be prepended to other image data, thus allowing the RPIX decoder to read an otherwise compliant image format. See Appendix E for more information.

In addition to its support for 8-bits-per-sample data, RPIX supports single-band monochrome images compressed using the FAX3 and FAX4 compression schemes.

When an RPIX image is decoded, only 1 or 3 bands are read. Which bands are selected can be determined by the image header or by the InputChannels operator. Similarly, *inter*Media writes only 1 or 3 band RPIX images.

RASF

extension: .ras

mime: image/x-ora-rasf

The Sun Raster image format, called RASF by *inter*Media, was developed by Sun Microsystems for its UNIX operating systems and has a wide distribution in the UNIX community. It supports a variety of pixel depths and includes support for a format-specific, run-length encoding compression scheme called SUNRLE by *inter*Media.

TGAF

extension: .tga

mime: image/x-ora-tgaf

The Truevision Graphics Adapter format (TGA, or TGAF to *inter*Media) was developed by Truevision, Inc., for their line of Targa and related graphics adapters. This format includes support for color images with 8, 16, 24, and 32 bits per pixel, and also includes support for a run-length encoding compression scheme called TARGARLE by *inter*Media.

TIFF

extension: .tif

mime: image/tiff

The Tag Image File Format (TIFF) was originally developed by the Aldus Corporation. The format has become something of a benchmark for image interchange and is extremely versatile, including support for a wide variety of compression and data formats, multiple image pages per file, and a wide variety of metadata. Because of its many options, TIFF is a good choice for many applications, including document storage, simple art, photographic and photo-realistic images, and others.

Oracle *inter*Media supports the "baseline TIFF" specification and also includes support for some TIFF "extensions," including tiled images and certain compression formats not included as part of the baseline TIFF specification. "Planar" TIFF images are not supported. It is important to note that the JPEG support in TIFF provided by *inter*Media is based on the revised JPEG in TIFF specification and not the original JPEG in TIFF specification. TIFF images in either big endian or little endian format can be read, but *inter*Media always writes big endian TIFFs.

Although the TIFF decoder in *inter*Media includes support for page selection using the "page" verb in the process( ) and processCopy( ) methods, the setProperties( ) method always returns the properties of the initial page in the file. It is important to note that this initial page is accessed by setting "page=0" in the process command string. Oracle *inter*Media currently does not support writing multiple page TIFF files.

WBMP

extension: .wbmp

mime: image/vnd.wap.wbmp

The Wireless Bitmap format (WBMP) was developed for the Wireless Application Protocol (WAP) as a means of transmitting bitmap (monochrome) images to WAP-compliant devices. An extremely minimalist format, it does not even include identifying markers or support for compression. It is most appropriate for very small images being transmitted over limited bandwidth networks.

The WBMP format is not related to the BMPF format.

B.2 Image Compression Formats

Image compression formats are listed alphabetically.

ASCII

Not an actual compression format by itself, ASCII is an encoding format used by PBM, PGM, and PPM images to represent images in plain ASCII text form. Each pixel value is represented by an individual integer in an ASCII-encoded PBM (or PGM or PPM) file.

BMPRLE

BMPRLE is the description that Oracle *inter*Media gives to images that are compressed with the BMP run-length encoding compression scheme. This compression format is available only for 4-bit and 8-bit LUT data, and only for images that are stored in INVERSE scanline order (the default order for BMP files). For very complex images, this compression can occasionally actually increase the file size.

DEFLATE

DEFLATE is the compression scheme employed by the PNG image format, and has also been adapted to work in the TIFF image format. DEFLATE is based on the

LZ77 algorithm (which is used in various zip utilities) and is a very adaptable compression scheme that handles a wide variety of image data formats well. Besides being used to compress image data in PNG and TIFF files, DEFLATE is also used within PNG files to compress some metadata.

DEFLATE-ADAM7

DEFLATE-ADAM7 is the same compression format as DEFLATE, but refers to images whose scanlines are interlaced for progressive display as the image is decoded. The intention of this technique is to allow a user to observe the image being progressively decoded as it is downloaded through a low bandwidth link, and quit before completion of the download. While the low bandwidth requirement is not typically relevant anymore, many existing images employ this encoding. Unlike JPEG-PROGRESSIVE and GIFLZW-INTERLACED, DEFLATE-ADAM7 interlaces images both horizontally and vertically.

Oracle *inter*Media provides read support for this encoding, but does not provide write support.

FAX3

FAX3 is the *inter*Media designation for CCITT Group 3 2D compression, which was developed by the CCITT (International Telegraph and Telephone Consultative Committee) as a protocol for transmitting monochrome images over telephone lines by facsimile and similar machines. The more official designation for this compression scheme is CCITT T.4.

Because this compression format supports only monochrome data, it cannot be used for color or grayscale images. This compression scheme uses a fixed dictionary that was developed using handwritten and typewritten documents and simple line graphics that were meant to be representative of documents being transmitted by facsimile. For this reason, although the compression can be used on images that have been dithered to monochrome, it may not produce as high a compression ratio as more adaptive schemes such as LZW or DEFLATE in those cases. FAX3 is most appropriate for scanned documents.

FAX4

FAX4 is the *inter*Media designation for CCITT Group 4 2D compression, which was developed by the CCITT (International Telegraph and Telephone Consultative Committee) as a protocol for transmitting monochrome images over telephone lines by facsimile and similar machines. The more official designation for this compression scheme is CCITT T.6.

Because this compression format supports only monochrome data, it cannot be used for color or grayscale images. This compression scheme uses a fixed dictionary that was developed using handwritten and typewritten documents and simple line graphics that were meant to be representative of documents being transmitted by facsimile. For this reason, although the compression can be used on images that have been dithered to monochrome, it may not produce as high a compression ratio as more adaptive schemes such as LZW or DEFLATE in those cases. FAX4 is most appropriate for scanned documents.

GIFLZW

GIFLZW is the *inter*Media designation for the LZW compression system used within GIF format images, and is different from LZW compression as used by other file formats. GIFLZW is an adaptive compression scheme that provides good compression for a wide variety of image data, although it is least effective on very complex images, such as photographs.

GIFLZW-INTERLACED

GIFLZW-INTERLACED is the same compression format as GIFLZW, but refers to images whose scanlines are interlaced for progressive display as the image is decoded. The intention of this technique is to allow a user to observe the image being progressively decoded as it is downloaded through a low bandwidth link, and quit before completion of the download. While the low bandwidth requirement is not typically relevant anymore, many existing images employ this encoding.

Oracle *inter*Media provides read support for this encoding, but does not provide write support.

HUFFMAN3

HUFFMAN3 is the *inter*Media designation for the Modified Huffman compression scheme used by the TIFF image format. This compression format is based on the CCITT Group 3 1D compression format, but is not an official CCITT standard compression format.

Because this compression format supports only monochrome data, it cannot be used for color or grayscale images. This compression scheme uses a fixed dictionary that was developed using handwritten and typewritten documents and simple line graphics that were meant to be representative of documents being transmitted by facsimile. For this reason, although the compression can be used on images that have been dithered to monochrome, it may not produce as high a compression ratio as more adaptive schemes such as LZW or DEFLATE in those cases. HUFFMAN3 is most appropriate for scanned documents.

JPEG

The JPEG compression format was developed by the Joint Photographic Experts Group for storing photographic and photo-realistic images. The JPEG compression format is very complex, but most images belong to a class called "baseline JPEG," which is a much simpler subset. Oracle *inter*Media supports only baseline JPEG compression.

The JPEG compression scheme is a lossy compression format; that is, images compressed using JPEG can never be reconstructed exactly. JPEG works by eliminating spatial and chromatic details that the eye will probably not notice. While JPEG can compress most data quite well, the results may include serious cosmetic flaws for images that are not photographic, such as monochrome or simple art. Other compression schemes are more appropriate for those cases (FAX formats or PNG and GIF). Also, the lossy nature of this compression scheme makes JPEG inappropriate for images that must be edited, but it is a good choice for finished images that must be compressed as tightly as possible for storage or transmission.

JPEG-PROGRESSIVE

This compression format is a variation of the JPEG compression format in which image scanlines are interlaced, or stored in several passes, all of which must be decoded to compute the complete image. This variant is intended to be used in low bandwidth environments where users can watch the image take form as intermediate passes are decoded, and terminate the image display if desired. While the low bandwidth requirement is not typically relevant anymore, this variant sometimes results in a smaller encoded image and is still popular. Oracle *inter*Media provides read, but not write, support for this encoding.

LZW

LZW is the *inter*Media designation for the LZW compression system used within TIFF format images, and is different from LZW compression as used by other file formats. TIFF LZW is an adaptive compression scheme that provides good compression for a wide variety of image data, although it is least effective on very complex images. TIFF LZW works best when applied to monochrome or 8-bit grayscale or LUT data; the TIFF method of applying LZW compression to other data formats results in much lower compression efficiency.

LZWHDIFF

LZWHDIFF is the description that *inter*Media gives to images employing the TIFF LZW compression system and also utilizing the TIFF horizontal differencing predictor. This scheme is a technique that can improve the compression ratios for

24-bit color and 8-bit grayscale images in some situations, without loss of data. It generally does not improve compression ratios for other image types.

NONE

This is the description that *inter*Media gives to image data that is not compressed.

PACKBITS

The Packbits compression scheme was developed by Apple Computer, Inc., as a simple byte-oriented, run-length encoding scheme for general use. This scheme is used by the PICT image format and has been adapted to work in TIFF images as well. Like other run-length encoding schemes, this compression can actually increase the data size for very complex images.

PCXRLE

PCXRLE is the description given by *inter*Media to images that are compressed using the PCX run-length encoding scheme. For very complex images, this compression can occasionally actually increase the file size.

RAW

Not an actual compression format by itself, RAW is encoding used by PBM, PGM, and PPM images to represent images in binary form (versus the plain text form employed by the ASCII encoding). The PBM documentation refers to this format as RAWBITS.

SUNRLE

SUNRLE is the description used within *inter*Media for the run-length encoding scheme used in Sun Raster images. For very complex images, this compression can occasionally actually increase the file size.

TARGARLE

TARGARLE is the description given by *inter*Media to images compressed using the run-length encoding scheme supported by the TGAF file format. For very complex images, this compression can occasionally actually increase the file size.

B.3 Summary of Image File Format and Image Compression Format

Table B–1 summarizes the input and output support provided for process( ) and setProperties( ) methods for image file formats relative to content format

characteristics, such as content format, interpretation, and color space. Table B–2 summarizes input and output support provided for process( ) and setProperties( ) methods for image file formats relative to compression format. Table B–3 summarizes input and output support provided for process( ) and setProperties( ) methods for other format-specific characteristics, such as pixel layout, channel order, pixel order, and scanline order.

The following abbreviations are used in Table B–1, Table B–2, and Table B–3:

- I = Input support is provided for process( ), processCopy( ), and setProperties( ) methods
- O = Output support is provided for process( ) and processCopy( ) methods
- - = Neither input nor output support is provided

Table B–1 I/O Support for Image File Content Format Characteristics

File Format	Content Format												
	1bitLUT (RGB& GRAY)	**4bitLUT (RGB& GRAY)**	**8bitLUT (RGB& GRAY)**	**8bitLUT (RGB& GRAY) A/T[1]**	**4bit direct GRAY**	**8bit direct GRAY**	**16 bit GRAY alpha**	**16bit direct RGB**	**24bit direct RGB**	**32bit direct RGBA**	**48bit direct RGB**	**64bit direct RGBA**	**Mono chrome**
BMPF	I O	I O	I O	-	-	-	-	I	I O	I	-	-	I O
CALS	-	-	-	-	-	-	-	-	-	-	-	-	I O
FPIX	-	-	-	-	-	I	-	-	I	-	-	-	-
GIFF[2]	I O	I O	I O	I O	-	-	-	-	-	-	-	-	I O
JFIF[3]	-	-	-	-	-	I O	-	-	I O	-	-	-	-
PBMF	-	-	-	-	-	-	-	-	-	-	-	-	I O
PCXF	I	I	I	-	-	-	-	-	I	-	-	-	I
PGMF	-	-	-	-	-	I O	-	-	-	-	-	-	-
PICT[4]	I	I	I O	-	-	I O	-	I	I O	-	-	-	I O
PNGF	I O	I O	I O	I O	I O	I O	I O	I	I O	I O	I	I	I O
PNMF[5]	-	-	-	-	-	O	-	-	O	-	-	-	O
PPMF	-	-	-	-	-	-	-	-	I O	-	-	-	-
RPIX[6]	-	-	-	-	-	I O	-	-	I O	-	-	-	I O
RASF	-	-	I O	-	-	I O	-	-	I O	-	-	-	I O

Table B–1 (Cont.) I/O Support for Image File Content Format Characteristics

File Format	Content Format												
	1bitLUT (RGB& GRAY)	4bitLUT (RGB& GRAY)	8bitLUT (RGB& GRAY)	8bitLUT (RGB& GRAY) A/T[1]	4bit direct GRAY	8bit direct GRAY	16 bit GRAY alpha	16bit direct RGB	24bit direct RGB	32bit direct RGBA	48bit direct RGB	64bit direct RGBA	Mono chrome
TGAF	-	-	I O	-	-	I O	-	I	I O	I	-	-	-
TIFF[7]	I O	I O	I O	-	I O	I O	I	I	I O	I O	I	I	I O
WBMP	-	-	-	-	-	-	-	-	-	-	-	-	I O

[1] RGB + Alpha, RGB + transparency, GRAY + Alpha, GRAY + transparency

[2] Animated GIFFs may not be encoded.

[3] Supports EXIF images.

[4] Vector and object graphics are not supported.

[5] PNMF format is supported as PBMF, PGMF, or PPMF; output will be PBMF, PGMF, or PPMF as appropriate.

[6] Can decode 1 or 3 bands from an *n*-band image; only 1 or 3 bands may be encoded.

[7] TIFF image file format also supports the following content formats as input or input/output as specified: Tiled data - input, Photometric interpretation - input/output, MSB - input/output, and LSB - input; Planar (BSQ) is not supported; both MSB and LSB ordered files may be decoded; decoded output is MSB.

Table B–2 I/O Support for Image File Compression Formats

File Format	Compression Format																				
	NONE	JPEG[1]	JPEG-PROGRESSIVE	BMPRLE	PCXRLE	SUNRLE	TARGARLE	GIFLZW	GIFLZW-INTERLACED	LZW	LZWHDIFF[2]	FAX3[3]	FAX4[3]	HUFFMAN3[3]	PACKBITS	DEFLATE	DEFLATE-ADAM7	ASCII	RAW	Quality Specification	
BMPF[4]	I O	-	-	I O	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	
CALS	-	-	-	-	-	-	-	-	-	-	-	-	I O	-	-	-	-	-	-	-	
FPIX	-	I	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	

Table B–2 (Cont.) I/O Support for Image File Compression Formats

File Format	Compression Format																				
	NONE	JPEG[1]	JPEG-PROGRESSIVE	BMPRLE	PCXRLE	SUNRLE	TARGARLE	GIFLZW	GIFLZW-INTERLACED	LZW	LZWHDIFF[2]	FAX3[3]	FAX4[3]	HUFFMAN3[3]	PACKBITS	DEFLATE	DEFLATE-ADAM7	ASCII	RAW	Quality Specification
GIFF	-	-	-	-	-	-	-	I O	I	-	-	-	-	-	-	-	-	-	-	-
JFIF[5]	-	I O	I	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	O
PBMF	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	I O	I O	-
PCXF	-	-	-	-	I	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
PGMF	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	I O	I O	-
PICT	-	I O	-	-	-	-	-	-	-	-	-	-	-	-	I O	-	-	-	-	-
PNGF	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	I O	I	-	-	-
PNMF	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	O	O	-
PPMF	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	I O	I O	-
RPIX	I O	-	-	-	-	-	-	-	-	-	-	I O	I O	-	-	-	-	-	-	-
RASF	I O	-	-	-	-	I O	-	-	-	-	-	-	-	-	-	-	-	-	-	-
TGAF	I O	-	-	-	-	-	I O	-	-	-	-	-	-	-	-	-	-	-	-	-
TIFF	I O	I O	-	-	-	-	-	-	-	I O	I O	I O	I O	I O	I O	I O	-	-	-	-
WBMP	I O	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-

[1] Supports 8-bit grayscale and 24-bit RGB data only.
[2] Supports 8-bit and 24-bit data only.
[3] Supports MONOCHROME data only.
[4] Compression is supported only for scanlineOrder=INVERSE (inverse DIB), which is the default.
[5] Supports EXIF images.

Table B–3 I/O Support for Image File Formats Other Than Content and Compression

File Format	Pixel Layout			Channel Order		Pixel Order			Scanline Order		Other Options		
	BIP	BIL	BSQ	RGB	RBG, GRB, GBR, BRG, BGR	NORMAL	REVERSE	OS/2	NORMAL	INVERSE	Input Channels	Page Selection	Tiled Data/ Tiled Output
BMPF	I O	-	-	I O	-	I O	-	I	I O	I O	-	-	-
CALS	I O	-	-	-	-	I O	-	-	I O	-	-	-	-
FPIX	I	-	-	I	-	I	-	-	I	-	-	-	-
GIFF[1]	I O	-	-	I O	-	I O	-	-	I O	-	-	-	-
JFIF[2]	I O	-	-	I O	-	I O	-	-	I O	-	-	-	-
PBMF	I O	-	-	-	-	I O	-	-	I O	-	-	-	-
PCXF	I	-	-	I	-	I	-	-	I	-	-	-	-
PGMF	I O	-	-	-	-	I O	-	-	I O	-	-	-	-
PICT[3]	I O	-	-	I O	-	I O	-	-	I O	-	-	-	-
PNGF	I O	-	-	I O	-	I O	-	-	I O	-	-	-	-
PNMF[4]	O	-	-	O	-	O	-	-	O	-	-	-	-
PPMF	I O	-	-	I O	-	I O	-	-	I O	-	-	-	-
RPIX[5]	I O	I O	I O	I O	I O	I O	I O	-	I O	I O	I	-	-
RASF	I O	-	-	I O	-	I O	-	-	I O	-	-	-	-
TGAF	I O	-	-	I O	-	I O	-	-	I O	-	-	-	-
TIFF[6]	I O	-	-	I O	-	I O	-	-	I O	-	-	I	I O
WBMP	I O	-	-	-	-	I O	-	-	I O	-	-	-	-

[1] Animated GIFFs may not be encoded.

[2] Supports EXIF images.

[3] Vector and object graphics are not supported.

[4] PNMF format is supported as PBMF, PGMF, or PPMF; output will be PBMF, PGMF, or PPMF as appropriate.

[5] Can decode 1 or 3 bands from an *n*-band image; only 1 or 3 bands may be encoded.

[6] TIFF image file format also supports the following content formats as input or input/output as specified: Tiled data - input, Photometric interpretation - input/output, MSB - input/output, and LSB - input; Planar (BSQ) is not supported; both MSB and LSB ordered files may be decoded; decoded output is MSB.

C

Video File and Compression Formats

The following sections describe the video file and compression formats supported by Oracle *inter*Media *("inter*Media"). To use these sections, find the data format you are interested in, and then determine the supported formats using the table in each section. For example, Table C–1 shows that *inter*Media supports Apple QuickTime 3.0 MOOV file format and a variety of compression formats from Cinepak to Motion-JPEG (Format B).

C.1 Apple QuickTime 3.0 Data Formats

The supported Apple QuickTime 3.0 data format, file extension, and MIME type are as follow:

- Data format: MOOV
- File extension: .mov
- MIME type: video/quicktime

Table C–1 lists the supported Apple QuickTime 3.0 data compression format names and compression format codes. The compression format codes are the FourCC codes that *inter*Media obtains from the dataFormat field of the video sample description entry of the 'stsd' atom in the QuickTime file. The table lists only the compression format codes recognized by *inter*Media.

Table C–1 Supported Apple QuickTime 3.0 Data Compression Formats

Compression Format Name	Compression Format Code
Cinepak	CVID
JPEG	JPEG
Uncompressed RGB	RGB

Table C–1 (Cont.) Supported Apple QuickTime 3.0 Data Compression Formats

Compression Format Name	Compression Format Code
Uncompressed YUV422	YUV2
Graphics	SMC
Animation: Run Length Encoded	RLE
Apple Video Compression	RPZA
Kodak Photo CD	KPCD
QuickDraw GX	QDGX
MPEG Still Image	MPEG
Motion-JPEG (Format A)	MJPA
Motion-JPEG (Format B)	MJPB

C.2 Microsoft Video for Windows (AVI) Data Formats

The following lists the supported Microsoft Video for Windows data format, file extension, and MIME type:

- Format: AVI
- File extension: .avi
- MIME type: video/x-msvideo

Table C–2 lists the supported Microsoft Video for Windows (AVI) compression format names and compression format codes. The compression format codes are the FourCC codes that *inter*Media obtains from the compression field of the 'strf' chunk in the AVI file. The table lists only the compression format codes recognized by *inter*Media.

Table C–2 Supported AVI Data Compression Formats

Compression Format Name	Compression Format Code
Microsoft Video 1	CRAM
Intel Indeo 3.1	IV31
Intel Indeo 3.2	IV32
Intel Indeo 4.0	IV40
Intel Indeo 4.1	IV41

Table C–2 (Cont.) Supported AVI Data Compression Formats

Compression Format Name	Compression Format Code
Intel Indeo 5.0	IV50
Intel Indeo 5.1	IV51
Cinepak	CVID

C.3 RealNetworks Real Video Data Format

The following lists the supported RealNetworks Real Video data format, file extension, and MIME type:

- Format: RMFF
- File extension: .rm
- MIME type: video/x-pn-realvideo

C.4 Supported Video MPEG Data Formats

The supported video MPEG formats are MPEG1, MPEG2, and MPEG4, as described in the following sections.

C.4.1 Supported MPEG1 and MPEG2 Data Formats

The following lists the supported video MPEG1 and MPEG2 data format, file extension, and MIME type:

- Format: MPEG
- File extension: .mpg
- MIME type: video/mpeg

C.4.2 Supported MPEG4 Data Formats

The following lists the supported video MPEG1 and MPEG2 data format, file extension, and MIME type:

- Format: MP4
- File extension: .mp4
- MIME type: application/mpeg4

D

Image process( ) and processCopy( ) Operators

This appendix describes the command options, or operators, used in the Oracle *inter*Media *("inter*Media") process( ) and processCopy( ) methods.

The available operators fall into three broad categories, each described in its own section:

- Section D.2, "Image Formatting Operators"
- Section D.3, "Image Processing Operators"
- Section D.4, "Format-Specific Operators"

Section D.1, "Common Concepts" describes the relative order of these operators.

> **Note:** Information about supported image file formats and image compression formats are presented in Appendix B. See Table B–1, Table B–2, and Table B–3, in particular.

D.1 Common Concepts

This section describes concepts common to all the image operators and the process( ) and processCopy( ) methods.

D.1.1 Source and Destination Images

The process( ) and processCopy( ) methods operate on one image, called the source image, and produce another image, called the destination image. In the case of the process( ) method, the destination image is written into the same storage space as

the source image, replacing it permanently. For the processCopy( ) method, the storage for the destination image is distinct from the storage for the source image.

D.1.2 process( ) and processCopy( )

The process( ) and processCopy( ) methods are functionally identical except for the fact that the process( ) method writes its output into the same BLOB from which it takes its input while the processCopy( ) method writes its output into a different BLOB. Their command string options are identical and no distinction is drawn between them.

For the rest of this appendix, the names process( ) and processCopy( ) are used interchangeably, and the use of the name process( ) implies both process( ) and processCopy( ) unless explicitly noted otherwise.

D.1.3 Operator and Value

Unless otherwise noted, the process( ) operators appear in the command string in the form <operator> = <value>. The right-hand side of the expression is called the **value** of the operator, and determines how the operator will be applied.

D.1.4 Combining Operators

In general, any number of operators can be combined in the command string passed into the process( ) method if the combination makes sense. However, certain operators are supported only if other operators are present or if other conditions are met. For example, the compressionQuality operator is supported only if the compression format of the destination image is JPEG. Other operators require that the source or destination image be a Raw Pixel or foreign image.

The flexibility in combining operators allows a single operation to change the format of an image, reduce or increase the number of colors, compress the data, and cut or scale the resulting image. This is highly preferable to making multiple calls to do each of these operations sequentially.

D.2 Image Formatting Operators

At the most abstract level, the image formatting operators are used to change the layout of the data within the image storage. They do not change the semantic content of the image, and unless the source image contains more information than the destination image can store, they do not change the visual appearance of the

image at all. Examples of a source image with more information than the destination image can store are:

- Converting a 24-bit image to an 8-bit image (too many bits per pixel)
- Converting a color image to a grayscale or monochrome image (too many color planes)
- Converting an uncompressed image, or an image stored in a lossless compression format, to a lossy compression format (too much detail)

D.2.1 fileFormat

The **fileFormat** operator determines the image file type, or format, of the output image. The value of this operator is a 4-character code, which is a mnemonic for the new file format name. The list of allowable values for the image fileFormat operator is shown in Table 6–1. Appendix B contains basic information about each file format, including its mnemonic (file format), typical file extension, allowable compression and content formats, and other notable features.

The value given to the fileFormat operator is the single most important detail when specifying the output for process( ). This value determines the range of allowable content and compression formats, whether or not compression quality will be useful, and whether or not the format-specific operators will be useful.

If the fileFormat operator is not used in the process( ) command string, *inter*Media will determine the file format of the source image and use that as the default file format value. If the file format of the source image does not support output, then an error will occur. If the source image is a foreign image, then the output image will be written as Raw Pixel.

D.2.2 contentFormat

The **contentFormat** operator determines the format of the image content. The content means the number of colors supported by the image and the manner in which they are supported. Depending on which file format is used to store the output image, some or most of the content formats may not be supported.

Image content formats fall into two broad categories, as follows:

- **Direct color (DRCT) images**

  In direct color images, the pixel data indicate color values directly, without reference to any additional information. This category includes monochrome

images (pure black and white), grayscale images (shades of gray) and RGB (true color) images.

In direct color images, the bit depth of the image indicates the size of the pixel data; monochrome images are implicitly 1 bit deep, grayscale images are 8 bits deep, or 16 if an optional 8-bit alpha channel is present, and RGB images are 24 bits deep -- usually 8 bits each for red, green, and blue, or 32 bits deep if an optional 8-bit alpha channel is present.

- **Lookup table (LUT) images**

  LUT images (also referred to as **indexed color images**) store possible color values in a table of possible color combinations, and pixel data then indicate which possible color from the table is to be used.

  The bit depth of a LUT image indicates both the size of the pixel data and the number of possible colors in the lookup table. A 1-bit LUT image would have 1-bit pixels and 2 possible colors (2^1), a 4-bit image would have 16 (2^4) possible colors, and an 8-bit image would have 256 (2^8) possible colors. Typically, the color table uses 24 bits to represent the possible colors, so although only 16 colors might be available in an image, they could each be any of up to 16 million possible RGB combinations. If the LUT image supports an alpha channel, then the table will usually use 32 bits to represent each color.

  If the contentFormat operator is not passed to the process( ) method, then *inter*Media attempts to duplicate the content format of the source image if it is supported by the file format of the destination image. Otherwise, a default content format is chosen depending on the destination file format.

The following four figures illustrate the syntax and options for the contentFormat operator.

Figure D–1 illustrates the contentFormat syntax that you use to convert an image to monochrome.

For finer control of the image output when you convert an image to monochrome, use the quantize operator with the ERRORDIFFUSION, ORDEREDDITHER, or THRESHOLD value. See Section D.3.7 on page D-11 for information on the quantize operator.

Figure D–1 Syntax Diagram for MONOCHROME contentFormat

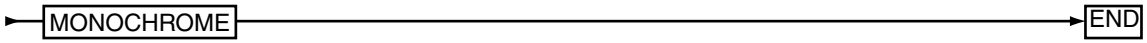


Figure D–2 illustrates the contentFormat syntax that you use to convert an image to LUT format.

The bit depth portion of the contentFormat syntax determines how many colors will be present in the LUT of the final image, as follows:

- An 8-bit image can contain up to 256 colors.
- A 4-bit image can contain up to 16 colors.
- A 1-bit image can contain only 2 colors, however, each of these colors may be any 24-bit RGB value.

The color portion of the contentFormat syntax controls whether the resulting image will be composed of RGB triplets or grayscale values. There is no difference between GRAY and GREY, and the optional SCALE suffix has no functional effect.

The A and T portion of the contentFormat syntax provides the ability to preserve alpha (A) or transparency (T) values in an image. You cannot use the transparency syntax to reduce a 32-bit image to an 8-bit image with alpha or transparency, but you can use it to preserve alpha or transparency when converting an image to a different file format. You can also use it to convert a transparency effect into a full alpha effect (however, only the transparent index will have alpha in the output).

For finer control of the image output when you convert a direct color image to a LUT color image, use the quantize operator with the ERRORDIFFUSION, ORDEREDDITHER, or MEDIANCUTvalue. See Section D.3.7 on page D-11 for information on the quantize operator.

Figure D–2 Syntax Diagram for LUT contentFormat

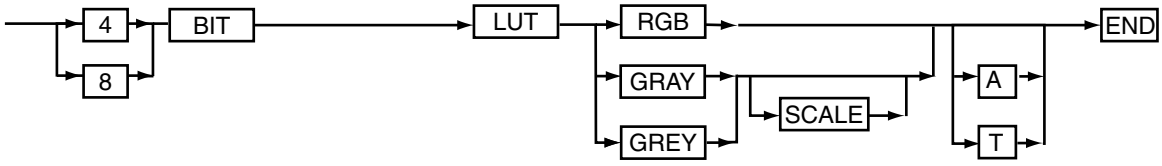


Figure D–3 illustrates the contentFormat syntax that you use to convert an image to grayscale.

The bit depth portion of the contentFormat syntax determines the overall type of the grayscale image: an 8-bit grayscale image may not have an alpha channel, while a 16-bit grayscale image currently must have an alpha channel. In either case, the DRCT specification is optional, because any non-LUT image will always be direct color. There is no difference between GRAY and GREY, and the optional SCALE

suffix has no functional effect. The alpha specification (A) is required for 16-bit grayscale output, and can be used to either preserve an existing alpha channel in a currently grayscale image or reduce a 32-bit RGBA image to grayscale with alpha.

The quantize operator has no effect on conversions to grayscale.

Figure D–3 Syntax Diagram for GRAYSCALE contentFormat

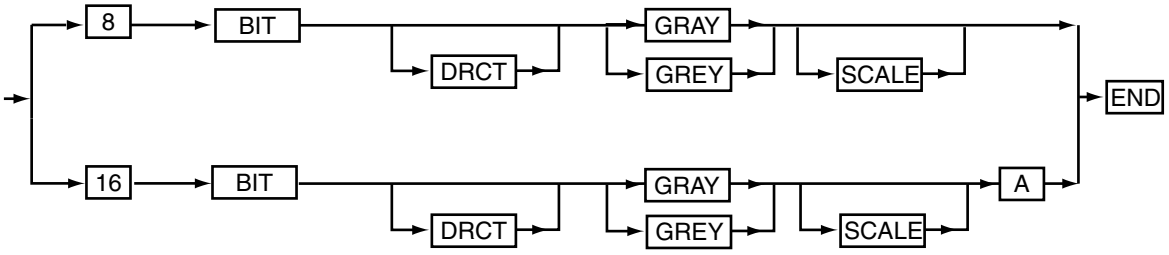


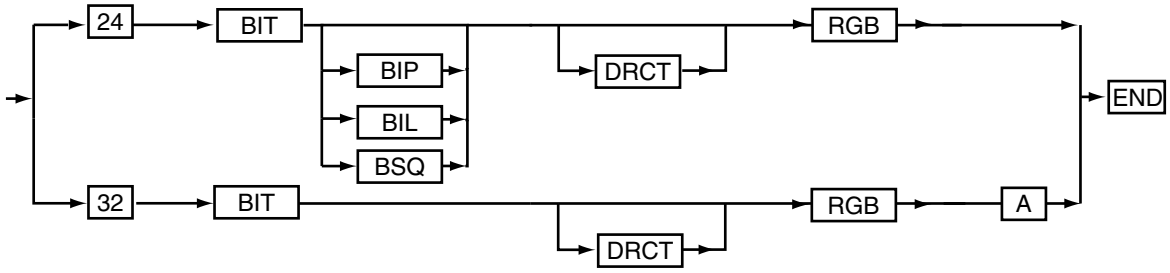
Figure D–4 illustrates the contentFormat syntax that you use to convert an image to direct color.

The bit depth portion of the contentFormat syntax determines the overall type of the direct RGB image: a 24-bit RGB image will not have an alpha channel, while a 32-bit RGB image must always have an alpha channel. In either case, the DRCT specification is optional because any non-LUT image will always be direct color. The alpha specification (A) is required for 32-bit RGB output; it preserves an existing alpha channel in a 32-bit or 64-bit RGB image, and it preserves the alpha channel in a 16-bit grayscale image that is being promoted to RGB.

The optional pixel chunking syntax allows images to be forced to band-interleaved-by-pixel (BIP, also known as chunky), band-interleaved-by-line (BIL), or band-interleaved-by-plane (BSQ, also known as band-sequential or planar). This portion of the syntax is supported only for RPIX formats.

The quantize operator is not used for conversions to direct color.

Figure D–4 Syntax Diagram for Direct RGB contentFormat



The following list of examples provides some common uses of the contentFormat operator:

- To specify that the output image be monochrome (black and white only):

```
image1.process('contentFormat=monochrome');
```

- To specify that the output image be an RGB lookup table (indexed color), either of the following is valid:

```
image1.process('contentFormat=8bitlutrgb');
image1.process('contentFormat=8bitlut');
```

- To specify that the output image be a grayscale lookup table (indexed color):

```
image1.process('contentFormat=8bitlutgray');
```

- To specify that the output image be grayscale, either of the following is valid:

```
image1.process('contentFormat=8bitgray');
image1.process('contentFormat=8bitgreyscale');
```

- To specify that the output image be direct color, either of the following is valid:

```
image1.process('contentFormat=24bitrgb');
image1.process('contentFormat=24bitdrctrgb');
```

- To specify that the output image be direct color and band sequential:

```
image1.process('contentFormat=24bitbsqrgb');
```

D.2.3 compressionFormat

The **compressionFormat** operator determines the compression algorithm used to compress the image data. The range of supported compression formats depends

heavily upon the file format of the output image. Some file formats support only a single compression format, and some compression formats are supported only by one file format.

The supported values for the compressionFormat operator are listed in Table 6–1.

All compression formats that include RLE in their mnemonic are run-length encoding compression schemes, and work well only for images that contain large areas of identical color. The PACKBITS compression type is a run-length encoding scheme that originates from the Macintosh system but is supported by other systems. It has limitations that are similar to other run-length encoding compression formats. Formats that contain LZW or HUFFMAN compression types are more complex compression schemes that examine the image for redundant information and are more useful for a broader class of images. FAX3 and FAX4 are the CCITT Group 3 and Group 4 standards for compressing facsimile data and are useful only for monochrome images. All the compression formats mentioned in this paragraph are **lossless** compression schemes, which means that compressing the image does not discard data. An image compressed into a lossless format and then decompressed will look the same as the original image.

The JPEG compression format is a special case. Developed to compress photographic images, the JPEG format is a **lossy** format, which means that it compresses the image typically by discarding unimportant details. Because this format is optimized for compressing photographic and similarly noisy images, it often produces poor results for other image types, such as line art images and images with large areas of similar color. JPEG is the only lossy compression scheme currently supported by *inter*Media.

The DEFLATE compression type is ZIP Deflate and is used by PNG image file formats. The DEFLATE-ADAM7 compression format is interlaced ZIP Deflate and is used by PNG image file formats. The ASCII compression type is ASCII encoding and the RAW compression type is binary encoding, and both are for PNM image file formats.

If the compressionFormat operator is not specified, then *inter*Media will use the default compression format; often this default is "None" or "No Compression."

If the compressionFormat operator is not specified and the file format of the destination image is different from that of the source image, then a default compression format will be selected depending on the destination image file format. This default compression is often "None" or "No Compression."

D.2.4 compressionQuality

The **compressionQuality** operator determines the relative quality of an image compressed with a lossy compression format. This operator has no meaning for lossless compression formats, and therefore is not currently supported for any compression format except JPEG.

The compressionQuality operator accepts integer values between 0 (lowest quality) and 100 (highest quality) in addition to five values, ranging from most compressed image (lowest visual quality) to least compressed image (highest visual quality): MAXCOMPRATIO, HIGHCOMP, MEDCOMP, LOWCOMP, and MAXINTEGRITY. Using the MAXCOMPRATIO value allows the image to be stored in the smallest amount of space but may introduce visible aberrations into the image. Using the MAXINTEGRITY value keeps the resulting image more faithful to the original but will require more space to store.

If the compressionQuality operator is not supplied and the destination compression format supports compression quality control, the quality will default to MEDCOMP for medium quality.

D.3 Image Processing Operators

The image processing operators supported by *inter*Media directly change the way the image looks on the display. The operators supported by *inter*Media represent only a fraction of all possible image processing operations, and are not intended for users performing intricate image analysis.

D.3.1 contrast

The **contrast** operator is used to adjust contrast. You can adjust contrast by percentage or by upper and lower bound, as follows:

- By percentage

  To adjust contrast by percentage, the syntax is as follows:

  ```
  contrast = <percent1> [<percent2> <percent3>]
  ```

  One or three parameters may be specified when specifying contrast by percentage. If one value is passed, then it is applied to all color components (either gray, or red, green, and blue) of the input image. If three values are specified then percent1 is applied to the red component of the image, percent2 to the green component, and percent3 to the blue component.

The percent values are floating-point numbers that indicate the percentage of the input pixel values that are mapped onto the full available output range of the image; the remaining input values are forced to either extreme (zero or full intensity). For example, a percentage of 60 indicates that the middle 60% of the input range is to be mapped to the full output range of the color space, while the lower 20% of the input range is forced to zero intensity (black for a grayscale image) and the upper 20% of the input range is forced to full intensity (white for a grayscale image).

- By upper and lower bound

  To adjust contrast by lower and upper bound, the syntax is as follows:

  ```
  contrast = <lower1> <upper1> [<lower2> <upper2> <lower3> <upper3>]
  ```

  The lower and upper values are integers that indicate the lower and upper bounds of the input pixel values that are to be mapped to the full output range. Values below the lower bound are forced to zero intensity and values above the upper bound are forced to full intensity. For 8-bit grayscale and 24-bit RGB images, these bounds may range from 0 to 255.

  Two or six values can be specified when using this contrast mode. If two values are specified, then those bounds are used for all color components of the image. If six values are specified, then lower1 and upper1 are applied to the red component of the image, lower2 and upper2 are applied to the green component, and lower3 and upper3 are applied to the blue component.

D.3.2 cut

The **cut** operator is used to create a subset of the original image. The values supplied to the cut operator are the origin coordinates (x,y) of the cut window in the source image, and the width and height of the cut window in pixels. This operator is applied before any scaling that is requested.

If the cut operator is not supplied, the entire source image is used.

D.3.3 flip

The **flip** operator places an image's scanlines in reverse order such that the scanlines are swapped from top to bottom. This operator accepts no values.

D.3.4 gamma

The **gamma** operator corrects the gamma (brightness) of an image. This operator accepts either one or three floating-point values using the following syntax:

```
gamma = <gamma1> [<gamma2> <gamma3>]
```

The values gamma1, gamma2, and gamma3 are the denominators of the gamma exponent applied to the input image. If only one value is specified, then that value is applied to all color components (either gray, or red, green, and blue) of the input image. If three values are specified then gamma1 is applied to the red component of the image, gamma2 to the green component, and gamma3 to the blue component.

To brighten an image, specify gamma values greater than 1.0; typical values are in the range 1.0 to 2.5. To darken an image, specify gamma values smaller than 1.0 (but larger than 0).

D.3.5 mirror

The **mirror** operator places an image's scanlines in inverse order such that the pixel columns are swapped from left to right. This operator accepts no values.

D.3.6 page

The **page** operator allows page selection from a multipage input image. The value specifies the input page that should be used as the source image for the process operation. The first page is numbered 0, the second page is 1, and so on.

Currently, only TIFF images support page selection.

D.3.7 quantize

The **quantize** operator affects the outcome of the contentFormat operator when you change the bit depth of an image. When an explicit change in content format is requested, or when the content format has to be changed due to other requested operations (such as scaling a LUT image, which requires promotion to direct color before scaling, or converting to a file format that only supports LUT images), the quantize operator indicates how any resulting quantization (reduction in number of colors) will be performed.

The value of the quantize operator can be any one of the following, referred to as quantizers:

- ERRORDIFFUSION

You can use the ERRORDIFFUSION quantizer in 2 ways: to reduce an 8-bit grayscale image to a monochrome image, or to reduce a 24-bit RGB image to an 8-bit LUT image.

The ERRORDIFFUSION quantizer retains the error resulting from the quantization of an existing pixel and diffuses that error among neighboring pixels. This quantization uses a fixed color table. The result looks good for most photographic images, but creates objectionable speckling artifacts for synthetic images. The artifacts are due to the fixed color lookup table used by the existing quantization method, which is statistically well balanced across the entire RGB color space, but is often a poor match for an image that contains many intensities of just a few colors. The result is more accurate than when the ORDEREDDITHER quantizer is specified; however, it is returned more slowly.

This is the default quantization value.

- ORDEREDDITHER

  You can use the ORDEREDDITHER quantizer in 2 ways: to reduce an 8-bit grayscale image to a monochrome image, or to reduce a 24-bit RGB image to an 8-bit LUT image.

  The ORDEREDDITHER quantizer finds the closest color match for each pixel in a fixed color table and then dithers the result to minimize the more obvious effects of color substitution. The result is satisfactory for most images but fine details can be lost in the dithering process. Although the result is not as accurate as when the ERRORDIFFUSION quantizer is specified, it is returned more quickly.

- THRESHOLD <threshold>

  The THRESHOLD quantizer reduces 8-bit grayscale images to monochrome images.

  The THRESHOLD quantizer assigns a monochrome output value (black or white) to a pixel by comparing that pixel's grayscale value to the threshold argument that is supplied along with the quantizer. If the input grayscale value is greater than or equal to the supplied threshold argument, then the output is white, otherwise the output is black. For an 8-bit grayscale or 24-bit RGB image, a grayscale value of 255 denotes white, while a grayscale value of 0 denotes black.

  For example, a threshold argument of 128 will cause any input value less than 128 to become black, while the remainder of the image will become white. A threshold value of 0 will cause the entire image to be white, and a value of 256

will cause the entire image to be black (for an 8-bit grayscale or a 24-bit RGB input image).

The THRESHOLD quantizer is most appropriately applied to synthetic images. The ERRORDIFFUSION and ORDEREDDITHER quantizers will produce better output when converting photographic images to monochrome, but will result in fuzziness in synthetic images; using the THRESHOLD quantizer will eliminate this fuzziness at the cost of the ability to discriminate between various intensities in the input image.

- MEDIANCUT [optional sampling rate]

  The MEDIANCUT quantizer reduces 24-bit RGB images to 8-bit LUT images.

  The MEDIANCUT quantizer generates a more optimal color table than the ERRORDIFFUSION or ORDEREDDITHER quantizers for some images, including most synthetic images, by choosing colors according to their popularity in the original image. However, the analysis of the original image is time consuming for large images, and some photographic images may look better when quantized using ERRORDIFFUSION or ORDEREDDITHER.

  The MEDIANCUT quantizer accepts an optional integer argument that specifies the sampling rate to be used when scanning the input image to collect statistics on color use. The default value for this quantizer argument is 1, meaning that every input pixel is examined, but any value greater than 1 may be specified. For a sampling rate *n* greater than 1, 1 pixel out of every *n* pixels is examined.

The following examples demonstrate how values and arguments are specified for the quantize operator:

```
image.process('contentformat=8bitlutrbg quantize = mediancut 2');
image.process('contentformat=monochrome quantize = threshold 128');
```

D.3.8 rotate

The **rotate** operator rotates an image within the image plane by the angle specified.

The value specified must be a floating-point number. A positive value specifies a clockwise rotation. A negative value for the operator specifies a counter-clockwise rotation. After the rotation, the image content is translated to an origin of 0,0 and the pixels not covered by the rotated image footprint are filled with the resulting colorspace black value.

Rotation values of 90, 180, and 270 use special code that quickly copies pixels without geometrically projecting them, for faster operation.

D.3.9 Scaling Operators

Oracle *inter*Media supports several operators that change the scale of an image, as described in the following sections.

D.3.9.1 fixedScale

The **fixedScale** operator is intended to simplify the creation of images with a specific size, such as thumbnail images. The scale, xScale, and yScale operators all accept floating-point scaling ratios, while the fixedScale (and maxScale) operators specify scaling values in pixels.

The two integer values supplied to the fixedScale operator are the desired dimensions (width and height) of the destination image. The supplied dimensions may be larger or smaller (or one larger and one smaller) than the dimensions of the source image.

The scaling method used by this operator will be the same as used by the scale operator in all cases. This operator cannot be combined with other scaling operators.

D.3.9.2 maxScale

The **maxScale** operator is a variant of the fixedScale operator that preserves the aspect ratio (relative width and height) of the source image. The maxScale operator also accepts two integer dimensions, but these values represent the maximum value of the appropriate dimension after scaling. The final dimension may actually be less than the supplied value.

Like the fixedScale operator, this operator is also intended to simplify the creation of images with a specific size. The maxScale operator is even better suited to thumbnail image creation than the fixedScale operator because thumbnail images created using the maxScale operator will have the same aspect ratio as the original image.

The maxScale operator scales the source image to fit within the dimensions specified while preserving the aspect ratio of the source image. Because the aspect ratio is preserved, only one dimension of the destination image may actually be equal to the values supplied to the operator. The other dimension may be smaller than, or equal to, the supplied value. Another way to think of this scaling method is that the source image is scaled by a single scale factor that is as large as possible, with the constraint that the destination image fit entirely within the dimensions specified by the maxScale operator.

If the cut operator is used in conjunction with the maxScale operator, then the aspect ratio of the cut window is preserved instead of the aspect ratio of the input image.

The scaling method used by this operator is the same as used by the scale operator in all cases. This operator cannot be combined with other scaling operators.

D.3.9.3 scale

The **scale** operator enlarges or reduces the image by the ratio given as the value for the operator. If the value is greater than 1.0, then the destination image will be scaled up (enlarged). If the value is less than 1.0, then the output will be scaled down (reduced). A scale value of 1.0 has no effect, and is not an error. No scaling is applied to the source image if the scale operator is not passed to the process( ) method.

There are two scaling techniques used by *inter*Media. The first technique is "scaling by sampling," and is used only if the requested compression quality is MAXCOMPRATIO or HIGHCOMP, or if the image is being scaled up in both dimensions. This scaling technique works by selecting the source image pixel that is closest to the pixel being computed by the scaling algorithm and using the color of that pixel. This technique is faster, but results in a poorer quality image.

The second scaling technique is "scaling by averaging," and is used in all other cases. This technique works by selecting several pixels that are close to the pixel being computed by the scaling algorithm and computing the average color. This technique is slower, but results in a better quality image.

If the scale operator is not used, the default scaling value is 1.0. This operator cannot be combined with other scaling operators.

D.3.9.4 xScale

The **xScale** operator is similar to the scale operator but affects only the width (x-dimension) of the image. The important difference between xScale and scale is that with xScale, scaling by sampling is used whenever the image quality is specified to be MAXCOMPRATIO or HIGHCOMP, and is not dependent on whether the image is being scaled up or down.

This operator may be combined with the yScale operator to scale each axis differently. It may not be combined with other scaling operators (Scale, fixedScale, maxScale).

D.3.9.5 yScale

The **yScale** operator is similar to the scale operator but affects only the height (y-dimension) of the image. The important difference between yScale and scale is that with yScale, scaling by sampling is used whenever the image quality is specified to be MAXCOMPRATIO or HIGHCOMP, and is not dependent on whether the image is being scaled up or down.

This operator may be combined with the xScale operator to scale each axis differently. It may not be combined with other scaling operators (scale, fixedScale, maxScale).

D.3.10 tiled

The **tiled** operator forces the output image to be tiled and can be used only with TIFF file format images. The resulting tile size will depend on the compression format that you select.

D.4 Format-Specific Operators

The following operators are supported only when the destination image file format is Raw Pixel or BMPF (scanlineOrder operator only), with the exception of the inputChannels operator, which is supported only when the source image is Raw Pixel or a foreign image. It does not matter if the destination image format is set to Raw Pixel or BMPF explicitly using the fileFormat operator, or if the Raw Pixel or BMPF format is selected by *inter*Media automatically, because the source format is Raw Pixel, BMPF, or a foreign image.

D.4.1 channelOrder

The **channelOrder** operator determines the relative order of the red, green, and blue channels (bands) within the destination Raw Pixel image. The order of the characters R, G, and B within the mnemonic value passed to this operator determine the order of these channels within the output. The header of the Raw Pixel image will be written such that this order is not lost.

For more information about the Raw Pixel file format and the ordering of channels in that format, see Appendix E.

D.4.2 pixelOrder

The **pixelOrder** operator controls the direction of pixels within a scanline in a Raw Pixel Image. The value Normal indicates that the leftmost pixel of a scanline will

appear first in the image data stream. The value Reverse causes the rightmost pixel of the scanline to appear first.

For more information about the Raw Pixel file format and pixel ordering, see Appendix E.

D.4.3 scanlineOrder

The **scanlineOrder** operator controls the order of scanlines within a Raw Pixel or BMPF image. The value Normal indicates that the top display scanline will appear first in the image data stream. The value Inverse causes the bottom scanline to appear first. For BMPF, scanlineOrder = inverse is the default and ordinary value.

For more information about the Raw Pixel or BMPF file format and scanline ordering, see Appendix E.

D.4.4 inputChannels

As stated in Section D.4, the **inputChannels** operator is supported only when the source image is in Raw Pixel format, or if the source is a foreign image.

The inputChannels operator assigns individual bands from a multiband image to be the red, green, and blue channels for later image processing. Any band within the source image can be assigned to any channel. If desired, only a single band may be specified and the selected band will be used as the grayscale channel, resulting in a grayscale output image. The first band in the image is number 1, and the band numbers passed to the Input Channels operator must be greater than or equal to one, and less than or equal to the total number of bands in the source image. Only the bands selected the by inputChannels operator are written to the output. Other bands are not transferred, even if the output image is in Raw Pixel format.

It should be noted that every Raw Pixel or foreign image has these input channel assignments written into its header block, but that this operator overrides those default assignments.

For more information about the Raw Pixel file format and input channels, see Appendix E.

E

Image Raw Pixel Format

This appendix describes the Oracle Raw Pixel image format and is intended for developers and advanced users who wish to use the Raw Pixel format to import unsupported image formats into Oracle *inter*Media *("inter*Media"), or as a means to directly access the pixel data in an image.

Much of this appendix is also applicable to foreign images.

E.1 Raw Pixel Introduction

Oracle *inter*Media supports many popular image formats suitable for storing artwork, photographs, and other images in an efficient, compressed way, and provides the ability to convert between these formats. However, most of these formats are proprietary to at least some degree, and the format of their content is often widely variable and not suited for easy access to the pixel data of the image.

The Raw Pixel format is useful for applications that need direct access to the pixel data without the burden of the complex computations required to determine the location of pixels within a compressed data stream. This simplifies reading the image for applications that are performing pixel-oriented image processing, such as filtering and edge detection. This format is even more useful to applications that need to write data back to the image. Because changing even a single pixel in a compressed image can have implications for the entire image stream, providing an uncompressed format enables applications to write pixel data directly, and later compress the image with a single process( ) command.

This format is also useful to users who have data in a format not directly supported by *inter*Media, but is in a simple, uncompressed format. These users can prepend a Raw Pixel identifier and header onto their data and import it into *inter*Media. For users who need only to read these images (such as for import or conversion), this

capability is built into *inter*Media as "Foreign Image Support." How this capability is related to the Raw Pixel format is described in Section E.10.

In addition to supporting image types not already built into *inter*Media, the Raw Pixel format also permits the interpretation of N-band imagery, such as satellite images. Using Raw Pixel, one or three bands of an N-band image may be selected during conversion to another image format, allowing easy visualization within programs that do not otherwise support N-band images. Note that images written with the Raw Pixel format still may have only one or three bands.

The current version of the Raw Pixel format is 1.0. This appendix is applicable to Raw Pixel images of this version only, as the particulars of the format may change with other versions.

E.2 Raw Pixel Image Structure

A Raw Pixel image consists of a 4-byte image identifier, followed by a 30-byte image header, followed by an arbitrary gap of 0 or more bytes, followed by pixel data.

It is worth noting that Raw Pixel images are never color-mapped, and therefore do not contain color lookup tables.

The Raw Pixel header consists of the Image Identifier and the Image Header. The Image Header is actually composed of several fields.

Note that the first byte in the image is actually offset 0. All integer fields are unsigned and stored in big endian byte order.

Table E–1 describes the raw pixel image header structure.

Table E–1 Raw Pixel Image Header Structure

Name	Byte(s)	Description
Image Identifier	0:3	4-byte character array containing ASCII values for RPIX. This array identifies the image as a Raw Pixel image.
Image Header Length	4:7	Length of this header in bytes, excluding the identifier field. The value of this field may be increased to create a gap between the header fields and the pixel data in the image.
Major Version	8	Major version number of the Raw Pixel format used in the image.

Table E–1 (Cont.) Raw Pixel Image Header Structure

Name	Byte(s)	Description
Minor Version	9	Minor version number of the Raw Pixel format used in the image.
Image Width	10:13	Width of the image in pixels.
Image Height	14:17	Height of the image in pixels.
Compression Type	18	Compression type of the image: None, CCITT FAX Group 3, or CCITT FAX Group 4.
Pixel Order	19	Pixel order of the image: Normal or Reverse.
Scanline Order	20	Scanline order of the image: Normal or Inverse.
Interleave	21	Interleave type of the image: BIP, BIL, or BSQ.
Number of Bands	22	Number of bands in the image. Must be in the range 1 to 255.
Red Channel Number	23	The band number of the channel to use as a default for red. This field is the grayscale channel number if the image is grayscale.
Green Channel Number	24	The band number of the channel to use as a default for green. This field is zero if the image is grayscale.
Blue Channel Number	25	The band number of the channel to use as a default for blue. This field is zero if the image is grayscale.
Reserved Area	26:33	Not currently used. All bytes *must* be zero.

E.3 Raw Pixel Header Field Descriptions

This section describes the fields of the Raw Pixel header in greater detail.

Image Identifier

Occupying the first 4 bytes of a Raw Pixel image, the identifier string must always be set to the ASCII values "RPIX" (hex 52 50 49 58). These characters identify the image as being encoded in RPIX format.

This string is currently independent of the Raw Pixel version.

Image Header Length

The Raw Pixel reader uses the value stored in this field to find the start of the pixel data section within a Raw Pixel image. To find the offset of the pixel data in the image, the reader adds the length of the image identifier (always 4) to the value in the image header length field. Thus, for Raw Pixel 1.0 images with no post-header gap, the pixel data starts at offset 34.

For Raw Pixel version 1.0 images, this field normally contains the integer value 30, which is the length of the Raw Pixel image header (not including the image identifier). However, the Raw Pixel format allows this field to contain any value equal to or greater than 30. Any information in the space between the end of the header data and the start of the pixel data specified by this header length is ignored by the Raw Pixel reader. This is useful for users who wish to prepend a Raw Pixel header onto an existing image whose pixel data area is compatible with the Raw Pixel format. In this case, the header length would be set to 30 plus the length of the existing header. The maximum length of this header is 4,294,967,265 bytes (the maximum value that can be stored in the 4-byte unsigned field minus the 30-byte header required by the Raw Pixel format). This field is stored in big endian byte order.

Major Version

A single-byte integer containing the major version number of the Raw Pixel format version used to encode the image. The current Raw Pixel version is 1.0, therefore this field is 1.

Minor Version

A single-byte integer containing the minor version number of the Raw Pixel format version used to encode the image. The current Raw Pixel version is 1.0, therefore this field is 0.

Image Width

The width (x-dimension) of the image in pixels.

Although this field is capable of storing an image dimension in excess of 4 billion pixels, limitations within *inter*Media require that this field be a value between and 32767, inclusive. This field is stored in big endian byte order.

Image Height

The height (y-dimension) of the image in pixels.

Although this field is capable of storing an image dimension in excess of 4 billion pixels, limitations within *inter*Media require that this field be a value between 1 and 32767, inclusive. This field is stored in big endian byte order.

Compression Type

This field contains the compression type of the Raw Pixel image. This field may contain the following values:

Value	Name	Compression
1	NONE	No compression
2	FAX3	CCITT Group 3 compression
3	FAX4	CCITT Group 4 compression

For grayscale, RGB, and N-band images, the image is always uncompressed, and only a value of 0 is valid. If the compression type is value 1 or 2, then the image is presumed to be monochrome. In this case, the image is presumed to contain only a single band, and must specify normal pixel order, normal scanline order, and BIP interleave.

Pixel Order

This field describes the pixel order within the Raw Pixel image. Typically, pixels in a scanline are ordered from left to right, along the traditional positive x-axis. However, some applications require that scanlines be ordered from right to left.

This field may contain the following values:

Value	Name	Pixel Order
1	NORMAL	Leftmost pixel first
2	REVERSE	Rightmost pixel first

This field cannot contain 0, as this indicates an unspecified pixel order; this would mean the image could not be interpreted. For images with CCITT G3 and G4 compression types, this field must contain the value 1.

Scanline Order

This field describes the scanline order within the Raw Pixel image. Typically, scanlines in an image are ordered from top to bottom. However, some applications require that scanlines are ordered from bottom to top.

This field may contain the following values:

Value	Name	Scanline Order
1	NORMAL	Topmost scanline first
2	INVERSE	Bottommost scanline first

This field cannot contain 0, as this indicates an unspecified scanline order; this would mean the image could not be interpreted. For images with CCITT G3 and G4 compression types, this field must contain the value 1.

Interleave

This field describes the interleaving of the various bands within a Raw Pixel image. For more information on the meaning of the various interleave options, see Section E.5.3.

This field may contain the following values:

Value	Name	Interleave
1	BIP	Band Interleave by Pixel, or "chunky"
2	BIL	Band Interleave by Line
3	BSQ	Band SeQuential, or "planar"

This field cannot contain 0, as this indicates an unspecified interleave; this would mean the image could not be interpreted. For images with CCITT G3 and G4 compression types, this field must contain the value 1.

Number of Bands

This field contains the number of bands or planes in the image, and must be a value between 1 and 255, inclusive. This field may not contain the value 0.

For CCITT images, this field must contain the value 1.

Red Channel Number

This field contains the number of the band that is to be used as the red channel during image conversion operations. This may be used to change the interpretation of a normal RGB image, or to specify a default band to be used as red in an N-band image. This default may be overridden using the inputChannels operator in the process( ) or processCopy( ) methods.

If the image has only one band, or only one band from an N-band image should be selected for display, then the band number should be encoded as the red channel. In this case, the green and blue channels should be set to 0.

This field may not contain the value 0; it must contain a value between 1 and the number of bands, inclusive.

Green Channel Number

This field contains the number of the band that is to be used as the green channel during image conversion operations. This may be used to change the interpretation of a normal RGB image, or to specify a default band to be used as green in an N-band image. This default may be overridden using the inputChannels operator in the process( ) or processCopy( ) method.

If the image has only one band, or only one band from an N-band image should be selected for display, then the band number should be encoded as the red channel. In this case, the green and blue channels should be set to 0.

This field may contain a value between 0 and the number of bands, inclusive.

Blue Channel Number

This field contains the number of the band that is to be used as the blue channel during image conversion operations. This may be used to change the interpretation of a normal RGB image, or to specify a default band to be used as blue in an N-band image. This default may be overridden using the inputChannels operator in the process( ) or processCopy( ) method.

If the image has only one band, or only one band from an N-band image should be selected for display, then the band number should be encoded as the red channel. In this case, the green and blue channels should be set to 0.

This field may contain a value between 0 and the number of bands, inclusive.

Reserved Area

The application of these 8 bytes titled Reserved Area is currently under development, but they are reserved even within Raw Pixel 1.0 images. These bytes must all be cleared to 0. Failure to do so will create undefined results.

E.4 Raw Pixel Post-Header Gap

Apart from the image identifier and the image header, Raw Pixel version 1.0 images contain an optional post-header gap, which precedes the actual pixel data. Unlike the reserved area of the image header, the bytes in this gap can contain any values you want. This is useful to store additional metadata about the image, which in some cases may be the actual image header from another file format.

However, because there is no standard for the information stored in this gap, take care when storing metadata in this area as other users may interpret this data differently. It is also worth noting that when a Raw Pixel image is processed, information stored in this gap is not copied to the destination image. In the case of the process( ) method, which writes its output to the same location as the input, the source information will be lost unless the transaction in which the processing took place is rolled back.

E.5 Raw Pixel Data Section and Pixel Data Format

The data section of a Raw Pixel image is where the actual pixel data of an image is stored; this area is sometimes called the **bitmap data**. This section describes the layout of the bitmap data.

For images using CCITT compression, the bitmap data area stores the raw CCITT stream with no additional header. The rest of this section applies only to uncompressed images.

Bitmap data in a Raw Pixel image is stored as 8-bit per plane, per pixel, direct color, packed data. There is no pixel, scanline, or band blocking or padding. Scanlines may be presented in the image as either topmost first, or bottommost first. Within a scanline, pixels may be ordered leftmost first, or rightmost first. All these options are affected by interleaving in a relatively straightforward way; see the sections that follow for examples.

E.5.1 Scanline Ordering

On the screen, an image may look like the following:

```
1111111111...
2222222222...
3333333333...
4444444444...
```

Each digit represents a single pixel; the value of the digit is the scanline that the pixel is on.

Generally, the scanline that forms the upper or topmost row of pixels is stored in the image data stream before lower scanlines. The preceding image would appear as follows in the bitmap data stream:

```
...1111111111...2222222222...3333333333...4444444444...
```

Note that the first scanline appears earlier than the remaining scanlines. The Raw Pixel format refers to this scanline ordering as normal.

However, some applications prefer that the bottommost scanline appear in the data stream first:

```
...4444444444...3333333333...2222222222...1111111111...
```

The Raw Pixel format refers to this scanline ordering as inverse.

E.5.2 Pixel Ordering

On the screen, a scanline of an image may look like the following:

```
...123456789...
```

Each digit represents a single pixel; the value of the digit is the column that the pixel is in.

Generally, the data that forms the leftmost pixels is stored in the image data stream before pixels toward the right. The preceding scanline would appear as follows in the bitmap data stream:

```
...123456789...
```

Note that the left pixel appears earlier than the remaining pixels. The Raw Pixel format refers to this pixel ordering as normal.

However, some applications prefer that the rightmost pixel appear in the data stream first:

```
...987654321...
```

The Raw Pixel format refers to this pixel ordering as reverse.

E.5.3 Band Interleaving

Band interleaving describes the relative location of different bands of pixel data within the image buffer.

Bands are ordered by their appearance in an image data stream, with 1 being the first band, *n* being the last band. Band 0 would indicate no band or no data.

Band Interleaved by Pixel (BIP), or *Chunky*

BIP, or *chunky*, images place the various bands or channels of pixel data sequentially by pixel, so that all data for one pixel is in one place. If the bands of the image are the red, green, and blue channels, then a BIP image might look like this:

```
scanline 1: RGBRGBRGBRGBRGBRGBRGB…
scanline 2: RGBRGBRGBRGBRGBRGBRGB…
scanline 3: RGBRGBRGBRGBRGBRGBRGB…
…
```

Band Interleaved by Line (BIL)

BIL images place the various bands of pixel data sequentially by scanline, so that data for one pixel is spread across multiple notional rows of the image. This reflects the data organization of a sensor that buffers data by scanline. If the bands of the image are the red, green, and blue channels, then a BIL image might look like this:

```
scanline 1: RRRRRRRRRRRRRRRRRRRRR…
            GGGGGGGGGGGGGGGGGGGG…
            BBBBBBBBBBBBBBBBBBBB…
scanline 2: RRRRRRRRRRRRRRRRRRRRR…
            GGGGGGGGGGGGGGGGGGGG…
            BBBBBBBBBBBBBBBBBBBB…
scanline 3: RRRRRRRRRRRRRRRRRRRRR…
            GGGGGGGGGGGGGGGGGGGG…
            BBBBBBBBBBBBBBBBBBBB…
…
```

Band Sequential (BSQ), or Planar

Planar images place the various bands of pixel data sequentially by bit plane, so that data for one pixel is spread across multiple planes of the image. This reflects the data organization of some video buffer systems, which control the different electron guns of a display from different locations in memory. If the bands of the image are the red, green, and blue channels, then a planar image might look like this:

```
plane 1: RRRRRRRRRRRRRRRRR…  (part of scanline 1)
         RRRRRRRRRRRRRRRRR…  (part of scanline 2)
         RRRRRRRRRRRRRRRRR…  (part of scanline 3)
…
plane 2: GGGGGGGGGGGGGGGGG…  (part of scanline 1)
         GGGGGGGGGGGGGGGGG…  (part of scanline 2)
         GGGGGGGGGGGGGGGGG…  (part of scanline 3)
…
plane 3: BBBBBBBBBBBBBBBBB…  (part of scanline 1)
         BBBBBBBBBBBBBBBBB…  (part of scanline 2)
         BBBBBBBBBBBBBBBBB…  (part of scanline 3)
…
```

E.5.4 N-Band Data

The Raw Pixel format supports up to 255 bands of data in an image. The relative location of these bands of data in the image is described in Section E.5.3, which gives examples of interleaving for 3 bands of data.

In the case of a single band of data, there is no interleaving; all three schemes are equivalent. Examples of interleaving other numbers of bands are given in the following table. All images in the examples have three scanlines and four columns. Each band of each pixel is represented by a single-digit band number. Numbers that are unenclosed and are displayed in normal text represent the first scanline of the image, numbers that are enclosed in parentheses and are displayed in italic text represent the second scanline of the image, and numbers that are enclosed in brackets ([ ]) and are displayed in boldface text represent the third scanline of the image.

Bands	BIP	BIL	BSQ
2	12121212 *(12121212)* **[12121212]**	11112222 *(11112222)* **[11112222]**	1111*(1111)***[1111]** 2222*(2222)***[2222]**
4	1234123412341234 *(1234123412341234)* **[1234123412341234]**	1111222233334444 *(1111222233334444)* **[1111222233334444]**	1111*(1111)***[1111]** 2222*(2222)***[2222]** 3333*(3333)***[3333]** 4444*(4444)***[4444]**
5	12345123451234512345 *(12345123451234512345)* **[12345123451234512345]**	11112222333344445555 *(11112222333344445555)* **[11112222333344445555]**	1111*(1111)***[1111]** 2222*(2222)***[2222]** 3333*(3333)***[3333]** 4444*(4444)***[4444]** 5555*(5555)***[5555]**

E.6 Raw Pixel Header - C Language Structure

The following C language structure describes the Raw Pixel header in a programmatic way. This structure is stored unaligned in the image file (that is, fields are aligned on 1-byte boundaries) and all integers are stored in big endian byte order.

```
struct RawPixelHeader
{
unsigned char identifier[4]; /* Always "RPIX" */

unsigned longhdrlength; /* Length of this header in bytes */
/* Including the hdrlength field */
/* Not including the identifier field */
/* &k.hdrlength + k.hdrlength = pixels */

unsigned char majorversion; /* Major revision # of RPIX format */
unsigned char minorversion; /* Minor revision # of RPIX format */

unsigned long width;  /* Image width in pixels */
unsigned long height; /* Image height in pixels */
unsigned char comptype;   /* Compression (none, FAXG3, FAXG4, ... ) */
unsigned char pixelorder; /* Pixel order */
unsigned char scnlorder;  /* Scanline order */
unsigned char interleave; /* Interleaving (BIP/BIL/BSQ) */

unsigned char numbands; /* Number of bands in image (1-255) */
unsigned char rchannel; /* Default red channel assignment */
unsigned char gchannel; /* Default green channel assignment */
unsigned char bchannel; /* Default blue channel assignment */
/* Grayscale images are encoded in R */
/* The first band is 1, not 0 */
/* A value of 0 means "no band" */

unsigned char reserved[8]; /* For later use */
};
```

E.7 Raw Pixel Header - C Language Constants

The following C language constants define the values used in the Raw Pixel header:

```
#define RPIX_IDENTIFIER "RPIX"

#define RPIX_HEADERLENGTH 30
```

```
#define RPIX_MAJOR_VERSION 1
#define RPIX_MINOR_VERSION 0

#define RPIX_COMPRESSION_UNDEFINED 0
#define RPIX_COMPRESSION_NONE 1
#define RPIX_COMPRESSION_CCITT_FAX_G3 2
#define RPIX_COMPRESSION_CCITT_FAX_G4 3
#define RPIX_COMPRESSION_DEFAULT RPIX_COMPRESSION_NONE

#define RPIX_PIXEL_ORDER_UNDEFINED 0
#define RPIX_PIXEL_ORDER_NORMAL 1
#define RPIX_PIXEL_ORDER_REVERSE 2
#define RPIX_PIXEL_ORDER_DEFAULT RPIX_PIXEL_ORDER_NORMAL

#define RPIX_SCANLINE_ORDER_UNDEFINED 0
#define RPIX_SCANLINE_ORDER_NORMAL 1
#define RPIX_SCANLINE_ORDER_INVERSE 2
#define RPIX_SCANLINE_ORDER_DEFAULT RPIX_SCANLINE_ORDER_NORMAL

#define RPIX_INTERLEAVING_UNDEFINED 0
#define RPIX_INTERLEAVING_BIP 1
#define RPIX_INTERLEAVING_BIL 2
#define RPIX_INTERLEAVING_BSQ 3
#define RPIX_INTERLEAVING_DEFAULT RPIX_INTERLEAVING_BIP

#define RPIX_CHANNEL_UNDEFINED 0
```

Note that the various macros for the UNDEFINED values are meant to be illustrative and not necessarily used, except for "RPIX_CHANNEL_UNDEFINED," which is used for the green and blue channels of single-band images.

E.8 Raw Pixel PL/SQL Constants

The following PL/SQL constants define the values used in the raw pixel information. The constants represent the length of the RPIX image identifier plus the length of the RPIX header.

```
CREATE OR REPLACE PACKAGE ORDImageConstants AS
  RPIX_HEADER_LENGTH_1_0   CONSTANT INTEGER := 34;
END ORDImageConstants;
```

E.9 Raw Pixel Images Using CCITT Compression

Although the Raw Pixel format is generally aimed at uncompressed direct color images, provision is also made to store monochrome images using CCITT Fax Group 3 or Fax Group 4 compression. This is useful for storing scans of black and white pages, such as for document management applications. These images are generally impractical to store even as grayscale, as the unused data bits combined with the very high resolution used in these images would use excessive disk space.

Raw Pixels images using CCITT compression are treated as normal Raw Pixel images, with the following restrictions:

- The compression type field must contain the value 1 or 2 as outlined in Section E.3 (FAX3 or FAX4).
- The pixel order field must contain the value 1 (normal pixel order).
- The scanline order field must contain the value 1 (normal scanline order).
- The interleave field must contain the value 1 (BIP interleave).
- The number of bands field must contain the value 1 (one band).
- The red channel number field must contain the value 1.
- The green channel number and the blue channel number fields must contain the value 0 (no band).

In addition to these restrictions, applications that attempt to access pixel data directly will need to understand how to read and write the CCITT formatted data.

E.10 Foreign Image Support and the Raw Pixel Format

Oracle *inter*Media provides support for reading certain foreign images that can be described in terms of a few simple parameters, and whose data is arranged in a certain straightforward way within the image file. There is no list of the supported formats because the list would be very large and continually changing. Instead, there are some simple guidelines to determine if an image can be read using the foreign image support in *inter*Media. These rules are summarized in the following sections.

Header

Foreign images may have any header (or no header), in any format, as long as its length does not exceed 4,294,967,265 bytes. As has been noted before, all information in this header will be ignored.

Image Width

Foreign images may be up to 32,767 pixels wide.

Image Height

Foreign images may be up to 32,767 pixels high.

Compression Type

Foreign images must be uncompressed or compressed using CCITT Fax Group 3 or Fax Group 4. Other compression schemes, such as run-length encoding, are not currently supported.

Pixel Order

Foreign images may store pixels from left-to-right or right-to-left. Other pixel ordering schemes, such as boustrophedonic ordering, are not currently supported.

Scanline Order

Foreign images may have top-first or bottom-first scanline orders. Scanlines that are adjacent in the image display must be adjacent in the image storage. Some image formats stagger their image scanlines so that, for example, scanlines 1,5,9, and so forth, are adjacent, and then 2,6,10 are also adjacent. This is not currently supported.

Interleaving

Foreign images must use BIP, BIL, or BSQ interleaving. Other arrangements of data bands are not allowed, nor may bands have any pixel, scanline, or band-level blocking or padding.

Number of Bands

Foreign images may have up to 255 bands of data. If there are more bands of data, the first 255 can be accessed *if* the interleaving of the image is band sequential. In this case, the additional bands of data lie past the accessible bands and do not affect the layout of the first 255 bands. Images with other interleaving types may not have more than 255 bands because the additional bands will change the layout of the bitmap data.

Trailer

Foreign images may have an image trailer following the bitmap data, and this trailer may be of arbitrary length. However, such data is completely ignored by

*inter*Media, and there is no method (or need) to specify the presence or length of such a trailer.

If an image with such a trailer is modified with the process( ) or processCopy( ) methods, the resulting image will not contain this trailer. In the case of the processCopy( ) method, the source image will still be intact.

F

Exceptions and Error Messages

The following sections describe the Oracle *inter*Media (*"inter*Media") object exceptions. For information about the *inter*Media error messages, see *Oracle Database Error Messages*.

F.1 ORDAudioExceptions Exceptions

The following exceptions are associated with the ORDAudio object:

ORDAudioExceptions.AUDIO_DURATION_IS_NULL

Cause: This exception is raised when calling the getAudioDuration method and the duration is NULL.

Action: Set the duration for the audio object to a known value.

ORDAudioExceptions.AUDIO_ENCODING_IS_NULL

Cause: This exception is raised when calling the getEncoding method and the encoding is NULL.

Action: Set the encoding for the audio object to a known value.

ORDAudioExceptions.AUDIO_FORMAT_IS_NULL

Cause: This exception is raised when calling the getFormat method and the format is NULL.

Action: Set the format for the audio object to a known format.

ORDAudioExceptions.AUDIO_NUM_CHANNELS_IS_NULL

Cause: This exception is raised when calling the getNumberOfChannels method and the number of channels is NULL.

Action: Set the number of channels for the audio object to a known value.

ORDAudioExceptions.AUDIO_PLUGIN_EXCEPTION

Cause: This exception is raised when the audio plug-in raises an exception.

Action: Refer to the Oracle *inter*Media documentation for more information.

ORDAudioExceptions.AUDIO_SAMPLE_SIZE_IS_NULL

Cause: This exception is raised when calling the getSampleSize method and the sample size is NULL.

Action: Set the sample size for the audio object to a known value.

ORDAudioExceptions.AUDIO_SAMPLING_RATE_IS_NULL

Cause: This exception is raised when calling the getSamplingRate method and the sampling rate is NULL.

Action: Set the sampling rate for the audio object to a known value.

ORDAudioExceptions.DESCRIPTION_IS_NOT_SET

Cause: This exception is raised when calling the getDescription method and the description attribute is not set.

Action: Set the description attribute.

ORDAudioExceptions.INVALID_DESCRIPTION

Cause: This exception is raised when you call the setDescription( ) method with a value that is not valid.

Action: Set the value of the user_description parameter to an acceptable value.

ORDAudioExceptions.INVALID_MIME_TYPE

Cause: This exception is raised if the mime parameter value of the setMimeType method is NULL.

Action: Set the MIME parameter value to a known value.

ORDAudioExceptions.LOCAL_DATA_SOURCE_REQUIRED

Cause: This exception is raised if the data source is external.

Action: Set the source information to a local source.

ORDAudioExceptions.METHOD_NOT_SUPPORTED

Cause: This exception is raised when the method called is not supported.

Action: Do not call this method.

ORDAudioExceptions.NULL_INPUT_VALUE

Cause: This exception is raised if you call one of the set methods and the parameter value is NULL.

Action: Set the parameter to a known value.

ORDAudioExceptions.NULL_LOCAL_DATA

Cause: This exception is raised when source.localData is NULL.

Action: Initialize source.localData using an init method.

F.2 ORDDocExceptions Exceptions

The following exceptions are associated with the ORDDoc object:

ORDDocExceptions.DOC_PLUGIN_EXCEPTION

Cause: This exception is raised when the document plug-in raises an exception.

Action: Refer to the Oracle *inter*Media documentation for more information.

ORDDocExceptions.INVALID_MIME_TYPE

Cause: This exception is raised if the mime parameter value of the setMimeType method is NULL.

Action: Set the MIME parameter value to a known value.

ORDDocExceptions.INVALID_FORMAT_TYPE

Cause: This exception is raised if the knownFormat parameter value of the setFormat method is NULL.

Action: Set the FORMAT parameter value to a known value.

ORDDocExceptions.METHOD_NOT_SUPPORTED

Cause: This exception is raised when the method called is not supported.

Action: Do not call this method.

ORDDocExceptions.NULL_LOCAL_DATA

Cause: This exception is raised when source.localData is NULL.

Action: Initialize source.localData using an init method.

F.3 ORDImageExceptions Exceptions

The following exceptions are associated with the ORDImage object:

ORDImageExceptions.DATA_NOT_LOCAL

Cause: This exception is raised when the data is not local (the source.local attribute is 0.)

Action: Reset the source attribute information to a local image source. Call the import( ) or importFrom( ) method to import the data into the source.local attribute and set the source.local attribute to 1.

ORDImageExceptions.INVALID_MIME_TYPE

Cause: This exception is raised if the mime parameter value of the setMimeType method is NULL.

Action: Set the MIME parameter value to a known value.

ORDImageExceptions.NULL_CONTENT

Cause: This exception is raised when the image is NULL.

Action: Do not specify a NULL image.

ORDImageExceptions.NULL_DESTINATION

Cause: This exception is raised when the destination image is NULL.

Action: Pass an initialized destination image.

ORDImageExceptions.NULL_LOCAL_DATA

Cause: This exception is raised when source.localData is NULL.

Action: Initialize source.localData using an init method.

ORDImageExceptions.NULL_PROPERTIES_DESCRIPTION

Cause: This exception is raised when the description parameter to setProperties is not set.

Action: Set the description parameter if you are using a foreign image. Otherwise, do not pass the description parameter.

F.4 ORDVideoExceptions Exceptions

The following exceptions are associated with the ORDVideo object:

ORDVideoExceptions.DESCRIPTION_IS_NOT_SET

Cause: This exception is raised when calling the getDescription method and the description attribute is not set.

Action: Set the description attribute.

ORDVideoExceptions.INVALID_MIME_TYPE

Cause: This exception is raised if the mime parameter value of the setMimeType method is NULL.

Action: Set the MIME parameter value to a known value.

ORDVideoExceptions.LOCAL_DATA_SOURCE_REQUIRED

Cause: This exception is raised if the data source is external.

Action: Set the source information to a local source.

ORDVideoExceptions.NULL_INPUT_VALUE

Cause: This exception is raised if either the knownWidth or knownHeight parameter values of the setFrameSize method is NULL.

Action: Set these parameters to known values.

ORDVideoExceptions.NULL_LOCAL_DATA

Cause: This exception is raised when source.localData is NULL.

Action: Initialize source.localData using an init method.

ORDVideoExceptions.METHOD_NOT_SUPPORTED

Cause: This exception is raised when the method called is not supported.

Action: Do not call this method.

ORDVideoExceptions.VIDEO_FORMAT_IS_NULL

Cause: This exception is raised when calling the getFormat method and the format is NULL.

Action: Set the format for the video object to a known format.

ORDVideoExceptions.VIDEO_PLUGIN_EXCEPTION

Cause: This exception is raised when the video plug-in raises an exception.

Action: Refer to the Oracle *inter*Media documentation for more information.

F.5 ORDSourceExceptions Exceptions

The following exceptions are associated with the ORDSource object:

ORDSourceExceptions.EMPTY_SOURCE

Cause: This exception is raised when the value of the local attribute is 1 or NULL (TRUE), but the value of the localData attribute is NULL.

Action: Pass an initialized source.

ORDSourceExceptions.INCOMPLETE_SOURCE_INFORMATION

Cause: This exception is raised when the source information is incomplete or the value of the srcType attribute is NULL and the local attribute is neither 1 nor NULL.

Action: Check your source information and set srcType, srcLocation, or srcName attributes as needed.

ORDSourceExceptions.INCOMPLETE_SOURCE_LOCATION

Cause: This exception is raised when the value of srcLocation is NULL.

Action: Check your source location and set the srcLocation attribute.

ORDSourceExceptions.INCOMPLETE_SOURCE_NAME

Cause: This exception is raised when the value of srcName is NULL.

Action: Check your source name and set the srcName attribute.

ORDSourceExceptions.INVALID_SOURCE_TYPE

Cause: This exception is raised when you call a getBFile method and the value of the source.srcType attribute is other than "file".

Action: Ensure that the source type is "file".

ORDSourceExceptions.IO_ERROR

Cause: The *inter*Media FILE source plug-in was unable to write the BLOB contents to the specified operating system file.

Action: Check that the file directory path exists and can be accessed by Oracle Database. Check that the correct Java file permissions have been granted to the Oracle user.

ORDSourceExceptions.METHOD_NOT_SUPPORTED

Cause: This exception is raised when the method called is not supported by the source plug-in being used.

Action: Call a supported method.

ORDSourceExceptions.NULL_SOURCE

Cause: This exception is raised when the value of the localData attribute is NULL.

Action: Pass an initialized source.

ORDSourceExceptions.SOURCE_PLUGIN_EXCEPTION

Cause: This exception is raised when the source plug-in raises an exception.

Action: Refer to the Oracle *inter*Media documentation for more information.

F.6 ORDImageSIExceptions Exceptions

The following exceptions are associated with the Still Image objects:

ORDImageSIExceptions.ILLEGAL_HEIGHT_WIDTH_SPEC

Cause: The height or width parameter is NULL or is a negative value.

Action: Specify a positive value for the input height and width parameters.

ORDImageSIExceptions.NULL_CONTENT

Cause: The BLOB parameter was NULL.

Action: Specify a BLOB parameter that is not NULL.

ORDImageSIExceptions.UNSUPPORTED_IMAGE_FORMAT

Cause: The specified image format is not supported

Action: Invoke the method using a supported image format. Refer to the SI_INFORMTN_SCHEMA views and the Oracle *inter*Media documentation for more information.

G

Deprecated Audio and Video Methods

The following Oracle *inter*Media *("inter*Media") ORDAudio and ORDVideo *get* methods that accept a ctx parameter were deprecated in release 8.1.6:

ORDAudio

```
getFormat(ctx IN OUT RAW) RETURN VARCHAR2
getEncoding(ctx IN OUT RAW) RETURN VARCHAR2
getNumberOfChannels(ctx IN OUT RAW) RETURN INTEGER
getSamplingRate(ctx IN OUT RAW) RETURN INTEGER
getSampleSize(ctx IN OUT RAW) RETURN INTEGER
getCompressionType(ctx IN OUT RAW) RETURN VARCHAR2
getAudioDuration(ctx IN OUT RAW) RETURN INTEGER
```

ORDVideo

```
getFormat(ctx IN OUT RAW) RETURN VARCHAR2
getFrameSize(SELF IN OUT NOCOPY ORDVideo,
                            ctx IN OUT RAW,
                            retWidth OUT INTEGER,
                            retHeight OUT INTEGER)
getFrameResolution(ctx IN OUT RAW) RETURN INTEGER
getFrameRate(ctx IN OUT RAW) RETURN INTEGER
getVideoDuration(ctx IN OUT RAW) RETURN INTEGER
getNumberOfFrames(ctx IN OUT RAW) RETURN INTEGER
getCompressionType(ctx IN OUT RAW) RETURN VARCHAR2
getNumberOfColors(ctx IN OUT RAW) RETURN INTEGER
getBitRate(ctx IN OUT RAW) RETURN INTEGER
```

The following ORDAudio and ORDVideo comments methods were deprecated in release 9.0.1:

ORDAudio

```
-- Methods associated with the comments attribute
MEMBER PROCEDURE appendToComments(amount IN BINARY_INTEGER,
                                  buffer IN VARCHAR2),
MEMBER PROCEDURE writeToComments(offset IN INTEGER,
                                 amount IN BINARY_INTEGER,
                                 buffer IN VARCHAR2),
MEMBER FUNCTION readFromComments(offset IN INTEGER,
                                 amount IN BINARY_INTEGER := 32767)
               RETURN VARCHAR2,
PRAGMA RESTRICT_REFERENCES(readFromComments, WNDS, WNPS, RNDS, RNPS),

MEMBER FUNCTION locateInComments(pattern    IN VARCHAR2,
                                 offset     IN INTEGER := 1,
                                 occurrence IN INTEGER := 1)
                RETURN INTEGER,
MEMBER PROCEDURE trimComments(newlen IN INTEGER),
MEMBER PROCEDURE eraseFromComments(amount IN OUT NOCOPY INTEGER,
                                   offset IN INTEGER := 1),
MEMBER PROCEDURE deleteComments,
MEMBER PROCEDURE loadCommentsFromFile(fileobj  IN BFILE,
                                      amount   IN INTEGER,
                                      from_loc IN INTEGER := 1,
                                      to_loc   IN INTEGER := 1),
MEMBER PROCEDURE copyCommentsOut(dest     IN OUT NOCOPY CLOB,
                                 amount   IN INTEGER,
                                 from_loc IN INTEGER := 1,
                                 to_loc   IN INTEGER := 1),
MEMBER FUNCTION compareComments(
                  compare_with_lob        IN CLOB,
                  amount                  IN INTEGER := 4294967295,
                  starting_pos_in_comment IN INTEGER := 1,
                  starting_pos_in_compare IN INTEGER := 1)
               RETURN INTEGER,
PRAGMA RESTRICT_REFERENCES(compareComments, WNDS, WNPS, RNDS, RNPS),

MEMBER FUNCTION getCommentLength RETURN INTEGER,
PRAGMA RESTRICT_REFERENCES(getCommentLength, WNDS, WNPS, RNDS, RNPS),
```

ORDVideo

```
-- Methods associated with the comments attribute
MEMBER PROCEDURE appendToComments(amount IN BINARY_INTEGER,
                                  buffer IN VARCHAR2),
```

```
MEMBER PROCEDURE writeToComments(offset IN INTEGER,
                                 amount IN BINARY_INTEGER,
                                 buffer IN VARCHAR2),
MEMBER FUNCTION readFromComments(offset IN INTEGER,
                                 amount IN BINARY_INTEGER := 32767)
                RETURN VARCHAR2,
PRAGMA RESTRICT_REFERENCES(readFromComments, WNDS, WNPS, RNDS, RNPS),

MEMBER FUNCTION locateInComments(pattern    IN VARCHAR2,
                                 offset     IN INTEGER := 1,
                                 occurrence IN INTEGER := 1)
                 RETURN INTEGER,
MEMBER PROCEDURE trimComments(newlen IN INTEGER),
MEMBER PROCEDURE eraseFromComments(amount IN OUT NOCOPY INTEGER,
                                   offset IN INTEGER := 1),
MEMBER PROCEDURE deleteComments,
MEMBER PROCEDURE loadCommentsFromFile(fileobj  IN BFILE,
                                      amount   IN INTEGER,
                                      from_loc IN INTEGER :=1,
                                      to_loc   IN INTEGER :=1),
MEMBER PROCEDURE copyCommentsOut(dest     IN OUT NOCOPY CLOB,
                                 amount   IN INTEGER,
                                 from_loc IN INTEGER :=1,
                                 to_loc   IN INTEGER :=1),
MEMBER FUNCTION compareComments(
                   compare_with_lob        IN CLOB,
                   amount                  IN INTEGER := 4294967295,
                   starting_pos_in_comment IN INTEGER := 1,
                   starting_pos_in_compare IN INTEGER := 1)
                RETURN INTEGER,
PRAGMA RESTRICT_REFERENCES(compareComments, WNDS, WNPS, RNDS, RNPS),

MEMBER FUNCTION getCommentLength RETURN INTEGER,
PRAGMA RESTRICT_REFERENCES(getCommentLength, WNDS, WNPS, RNDS, RNPS),
```

The following ORDAudio and ORDVideo accessor methods were deprecated in release 9.0.1:

ORDAudio

```
MEMBER PROCEDURE setProperties(ctx IN OUT RAW),
```

ORDVideo

```
MEMBER PROCEDURE setProperties(ctx IN OUT RAW),
```

Index

A

B

C

D

E

F

G

H

I

J

L

M

N

O

P

Q

R

S

T

V

W

X

Y