# Oracle® Multimedia

Reference

11*g* Release 2 (11.2)

**E10776-03**

August 2010

Oracle Multimedia enables Oracle Database to store, manage, and retrieve images, audio, video, DICOM format medical images and other objects, or other heterogeneous media data in an integrated fashion with other enterprise information. Oracle Multimedia extends Oracle Database reliability, availability, and data management to multimedia content in Internet, electronic commerce, medical, financial, and other media-rich applications.

ORACLE®

Oracle Multimedia Reference, 11*g* Release 2 (11.2)

E10776-03

Copyright © 1999, 2010, Oracle and/or its affiliates. All rights reserved.

Primary Author: Sue Pelski

Contributors: Robert Abbott, Melliyal Annamalai, Fengting Chen, Dongbai Guo, Dong Lin, Susan Mavris, David Noblet, James Steiner, Yingmei Sun, Manjari Yalavarthy, Jie Zhang

This software and related documentation are provided under a license agreement containing restrictions on use and disclosure and are protected by intellectual property laws. Except as expressly permitted in your license agreement or allowed by law, you may not use, copy, reproduce, translate, broadcast, modify, license, transmit, distribute, exhibit, perform, publish, or display any part, in any form, or by any means. Reverse engineering, disassembly, or decompilation of this software, unless required by law for interoperability, is prohibited.

The information contained herein is subject to change without notice and is not warranted to be error-free. If you find any errors, please report them to us in writing.

If this software or related documentation is delivered to the U.S. Government or anyone licensing it on behalf of the U.S. Government, the following notice is applicable:

U.S. GOVERNMENT RIGHTS Programs, software, databases, and related documentation and technical data delivered to U.S. Government customers are "commercial computer software" or "commercial technical data" pursuant to the applicable Federal Acquisition Regulation and agency-specific supplemental regulations. As such, the use, duplication, disclosure, modification, and adaptation shall be subject to the restrictions and license terms set forth in the applicable Government contract, and, to the extent applicable by the terms of the Government contract, the additional rights set forth in FAR 52.227-19, Commercial Computer Software License (December 2007). Oracle USA, Inc., 500 Oracle Parkway, Redwood City, CA 94065.

This software is developed for general use in a variety of information management applications. It is not developed or intended for use in any inherently dangerous applications, including applications which may create a risk of personal injury. If you use this software in dangerous applications, then you shall be responsible to take all appropriate fail-safe, backup, redundancy, and other measures to ensure the safe use of this software. Oracle Corporation and its affiliates disclaim any liability for any damages caused by use of this software in dangerous applications.

Oracle is a registered trademark of Oracle Corporation and/or its affiliates. Other names may be trademarks of their respective owners.

This software and documentation may provide access to or information on content, products, and services from third parties. Oracle Corporation and its affiliates are not responsible for and expressly disclaim all warranties of any kind with respect to third-party content, products, and services. Oracle Corporation and its affiliates will not be responsible for any loss, costs, or damages incurred due to your access to or use of third-party content, products, or services.

# Contents

## 6 Oracle Multimedia ORDVideo Object Type

## 7 Oracle Multimedia Relational Interface

## 8 Oracle Multimedia ORDSource Object Type

## A Audio File and Compression Formats for Use With Oracle Multimedia



## B Image File and Compression Formats for Use With Oracle Multimedia



## C Video File and Compression Formats for Use With Oracle Multimedia



## D Oracle Multimedia Image Processing

## G Exceptions for Oracle Multimedia Objects



## H init( ) Methods for Oracle Multimedia Object Types



## I Oracle Multimedia SQL/MM Still Image Object Types

## J Oracle Multimedia Deprecated API Components



## Index

## List of Figures

## List of Tables

# Preface

This manual describes how to use Oracle Multimedia, which ships with Oracle Database. It provides detailed information about Oracle Multimedia APIs and their components, including object types and methods for multimedia data source manipulation.

In Oracle Database 11*g* Release 1 (11.1), the name Oracle *inter*Media was changed to Oracle Multimedia. The feature remains the same, only the name has changed. References to Oracle *inter*Media were replaced with Oracle Multimedia although, some references to Oracle *inter*Media or *inter*Media might still appear in graphical user interfaces, code examples, and related documents in the Documentation Library for Oracle Database 11*g* Release 2 (11.2).

The sample code in this manual might not match the code shipped with Oracle Database Examples media. To run examples that are shipped with Oracle Database Examples media on your system, use the files provided with Oracle Database Examples media. Do not attempt to compile and run the code in this guide.

See *Oracle Database New Features Guide* for information about Oracle Database and the features and options that are available to you.

## Audience

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

## Documentation Accessibility

Our goal is to make Oracle products, services, and supporting documentation accessible to all users, including users that are disabled. 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. Accessibility standards will continue to evolve over time, and Oracle 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 more information, visit the Oracle Accessibility Program Web site at `http://www.oracle.com/accessibility/`.

### Accessibility of Code Examples in Documentation

Screen readers 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, some screen readers 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 does not own or control. Oracle neither evaluates nor makes any representations regarding the accessibility of these Web sites.

### Access to Oracle Support

Oracle customers have access to electronic support through My Oracle Support. For information, visit `http://www.oracle.com/support/contact.html` or visit `http://www.oracle.com/accessibility/support.html` if you are hearing impaired.

## Related Documents

---

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

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

See your operating system-specific installation guide for more information.

---

For more information about using Oracle Multimedia in a development environment, see the following documents in the Oracle Database Online Documentation Library:

- *Oracle Multimedia User's Guide*
- *Oracle Multimedia DICOM Developer's Guide*
- *Oracle Call Interface Programmer's Guide*
- *Oracle Database Advanced Application Developer's Guide*
- *Oracle Database SecureFiles and Large Objects Developer's Guide*
- *Oracle Database Concepts*
- *Oracle Database PL/SQL Language Reference*
- *Oracle Database Error Messages*

For information about Oracle Locator, see *Oracle Spatial Developer's Guide*.

For more information about using XML, see *Oracle XML DB Developer's Guide*.

For reference information about Oracle Multimedia Java classes in Javadoc format, see the following Oracle API documentation (also known as Javadoc) in the Oracle Database Online Documentation Library:

- *Oracle Multimedia Java API Reference*
- *Oracle Multimedia Servlets and JSP Java API Reference*
- *Oracle Multimedia DICOM Java API Reference*

- *Oracle Multimedia Mid-Tier Java API Reference*

For information about using the Oracle Multimedia JSP Tag Library, see *Oracle Multimedia JSP Tag Library Guide* in the Oracle Multimedia Software section of the Oracle Technology Network Web site.

For more information about Java, including information about Java Advanced Imaging (JAI), see the API documentation provided by Oracle.

Many of the examples in this manual use the sample schemas. See *Oracle Database Sample Schemas* for information about how these schemas were created and how you can use them.

## Conventions

Although Boolean is a proper noun, it is presented as boolean in this manual when its use in Java code requires case-sensitivity.

The following text conventions are also used in this guide:

| Convention | Meaning |
|---|---|
| **boldface** | Boldface type indicates graphical user interface elements associated with an action, or terms defined in text or the glossary. |
| *italic* | Italic type indicates book titles, emphasis, or placeholder variables for which you supply particular values. |
| `monospace` | Monospace type indicates commands within a paragraph, URLs, code in examples, text that appears on the screen, or text that you enter. |

## Syntax Descriptions

Syntax descriptions are provided in this manual for various SQL, PL/SQL, or other command-line constructs in graphic form or Backus Naur Form (BNF). See *Oracle Database SQL Reference* for information about how to interpret these descriptions.

# What's New in Oracle Multimedia?

This document summarizes new features, enhancements, APIs, and Oracle Database support introduced with Oracle Multimedia in the current release. It also describes a feature that is no longer being enhanced, beginning with the current release.

## New Features for Release 11.2

Oracle Database 11*g* Release 2 (11.2) added the following new features, which are described in this manual.

**See Also:**

*Oracle Multimedia User's Guide* for the complete list of new features for Oracle Multimedia

*Oracle Multimedia DICOM Developer's Guide* for information about new DICOM enhancements

### New Constructor Functions

Oracle Multimedia now provides new constructor functions for BLOBs and other sources for these object types:

- ORDAudio

  See ORDAudio Constructors for complete reference information about these ORDAudio constructor functions: ORDAudio for BLOBS and ORDAudio for Other Sources.

- ORDDoc

  See ORDDoc Constructors for complete reference information about these ORDDoc constructor functions: ORDDoc for BLOBS and ORDDoc for Other Sources.

- ORDImage

  See ORDImage Constructors for complete reference information about these ORDImage constructor functions: ORDImage for BLOBS and ORDImage for Other Sources.

- ORDVideo

  See ORDVideo Constructors for complete reference information about these ORDVideo constructor functions: ORDVideo for BLOBS and ORDVideo for Other Sources.

As a result, the following init( ) methods were not enhanced in Oracle Database 11g, Release 2 (11.2):

- init( ) for ORDAudio
- init( ) for ORDDoc
- init( ) for ORDImage
- init( ) for ORDVideo
- init(srcType,srcLocation,srcName) for ORDAudio
- init(srcType,srcLocation,srcName) for ORDDoc
- init(srcType,srcLocation,srcName) for ORDImage
- init(srcType,srcLocation,srcName) for ORDVideo

---

**Note:**

These init( ) methods are still available in Release 11.2, but Oracle recommends taking these actions:

- Write new applications to use the new constructor functions.
- Migrate existing applications from the init( ) methods to the new constructor functions.

See Appendix H for more information about these methods.

---

### Image Watermarking

Oracle Multimedia now supports adding watermarks that contain images or text to source images, thus preventing the misuse of copyrighted or trademarked images. See Section D.6 for more information about this feature.

### New Image Processing Operators

Oracle Multimedia now provides these new image processing operators:

- nometadata

  Use the nometadata operator to process an image without encoding its image metadata in the resulting image. See Section D.3.6 for more information about this operator.

- sharpen

  Use the sharpen operator to improve the sharpness of an image. See Section D.3.11 for more information about this operator.

- transparencyFillColor

  Use the transparencyFillcolor operator to fill the transparent regions of PNG image file formats with color. See Section D.3.13 for more information about this operator.

### CMYK Content Format

Oracle Multimedia now supports the CMYK content format. See Section D.5 for more information about this feature.

## Status of ORDImage Support for DICOM in Release 11.2

Beginning with Oracle Database 11g, Release 2 (11.2), the DICOM support in ORDImage objects that was introduced in Oracle Database 10g, Release 2 (10.2) was not enhanced. The DICOM support in ORDImage objects is still available in Release 11.2, but Oracle recommends taking the following actions for medical imaging applications.

- For new applications:

  Use the new DICOM support that was introduced in Oracle Database 11g, Release 1 (11.1).

- For existing applications that use the DICOM support in ORDImage objects:

  Migrate to the new DICOM support that was introduced in Release 11.1.

See Section J.3 for more information about the status of this feature and its enhancements.

**See Also:**

*Oracle Multimedia DICOM Developer's Guide* for complete information about Oracle Multimedia DICOM features and enhancements

# 1

# Introduction to Oracle Multimedia

Oracle Multimedia (formerly Oracle *inter*Media) enables Oracle Database to store, manage, and retrieve images, DICOM format medical images and other objects, audio, video, or other heterogeneous media data in an integrated fashion with other enterprise information.

Oracle Multimedia extends Oracle Database reliability, availability, and data management to multimedia content in traditional, medical, Internet, electronic commerce, and media-rich applications.

This chapter includes these sections:



**See Also:**

- *Oracle Multimedia User's Guide* for conceptual information and information about application development
- *Oracle Multimedia DICOM Developer's Guide* for more information about Oracle Multimedia DICOM support

## 1.1 Multimedia Object Types and Methods

Oracle Multimedia provides the ORDAudio, ORDDoc, ORDImage, ORDVideo, and SI_StillImage object types and methods for:

- Extracting metadata and attributes from multimedia data
- Embedding metadata created by applications into image files
- Getting and managing multimedia data from Oracle Multimedia, Web servers, file systems, and other servers
- Performing manipulation operations on image data

Oracle Multimedia also provides the ORDDicom object type and methods for storing, managing, and manipulating DICOM format medical images and other data.

### SQL Syntax for Complex Objects

The object syntax for accessing attributes within a complex object is the dot notation (except in Java):

```
variable.data_attribute
```

The syntax for invoking methods of a complex object is also the dot notation (except in Java):

```
variable.function(parameter1, parameter2, ...)
```

In keeping with recommended programming practices, a complete set of media attribute accessors (get methods) and setters (set methods) are provided for accessing attributes for each media type.

**See Also:**

See *Oracle Database Concepts* for more information about complex objects

## 1.2 Multimedia Storage

Oracle Multimedia 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:** Do not call ORDSource methods 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.

See Chapter 8 for more information about ORDSource methods.

# 2

# Common Methods and Notes for Oracle Multimedia Object Types

This chapter provides reference and other information about the common methods used for these Oracle Multimedia object types:

- ORDAudio
- ORDDoc
- ORDImage
- ORDVideo

This chapter includes these sections:



**See Also:**

*Oracle Multimedia DICOM Developer's Guide* for information about the ORDDicom object type and methods for storing, managing, and manipulating DICOM format medical images and other data

## 2.1 Examples for Common Methods

The examples in this chapter use the ONLINE_MEDIA table in the Product Media (PM) sample schema. These examples assume that the table has been populated as shown in the examples in Chapter 3, Chapter 4, Chapter 5, and Chapter 6.

**Note:** The Oracle Multimedia methods are designed to be internally consistent. If you use Oracle Multimedia methods (such as import( ) or image process( )) to modify the media data, Oracle Multimedia ensures 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), you must ensure that the object attributes stay synchronized and the update time is modified; otherwise, the object attributes will not match the data.

**See Also:**

*Oracle Database Sample Schemas* for information about the PM and other sample schemas

## 2.2 Embedded ORDSource Object

The ORDSource object is embedded within the ORDAudio, ORDDoc, ORDImage, 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.

If the data is stored locally in a BLOB within Oracle Database, the localData attribute is used to find the media data, and the local flag indicates that the data is local. The srcType, srcLocation, and srcName attributes are not used.

If the data is stored externally in a BFILE, a URL, or a user-defined source, the srcType, srcLocation, and srcName attributes are used to find the media data, and the local flag indicates that the data is external.

See ORDSource Object Type for details on how the ORDSource object type is defined, including these ORDSource attributes:

- localData: the locally stored multimedia data stored as a BLOB within the object.
- srcType: the data source type. (See Section 2.2.1.)
- srcLocation: the place where data can be found based on the srcType value. (See Section 2.2.2.)
- srcName: the data object name. (See Section 2.2.3.)
- updateTime: the time at which the data was last updated.
- local: a flag that indicates whether the data is local or external.

---

**Note:** For HTTP sources, the srcLocation and srcName attributes are concatenated to construct a URL to locate the media object. For example:

If srcType is `HTTP`, srcLocation is `www.example.com/images/`, and srcName is `example.jpg`, then the URL to locate the media is `http://www.example.com/images/example.jpg`.

---

### 2.2.1 Definition of the srcType Attribute

The valid values for the srcType attribute are listed in Table 2–1.

***Table 2–1 srcType Values***

| Value | Description |
|---|---|
| `FILE` | A BFILE on a local file system |
| `HTTP` | An HTTP server |
| *<name>* | User-defined |

#### FILE

The srcType value `FILE` is a reserved word for the BFILE source plug-in provided by Oracle. To implement your own file plug-in, select a different name (for example: MYFILE).

#### HTTP

The srcType value `HTTP` is a reserved word for the HTTP source plug-in provided by Oracle.

### 2.2.2 Definition of the srcLocation Attribute

The valid values for the srcLocation attribute, for the corresponding srcType values, are listed in Table 2–2.

***Table 2–2 srcLocation Values for Corresponding srcType Values***

| srcType | Location Value |
|---|---|
| `FILE` | The name of the database directory object |
| `HTTP` | The base URL to locate the media directory (the prefix `http://` is not required) |
| *<name>* | An identifier string required to access a user-defined source |

### 2.2.3 Definition of the srcName Attribute

The valid values for the srcName attribute, for the corresponding srcType values, are listed in Table 2–3.

***Table 2–3 srcName Values for Corresponding srcType Values***

| srcType | Location Name |
|---|---|
| `FILE` | The name of the file |
| `HTTP` | The name of the object |
| *<name>* | The name of the object |

## 2.3 Important Notes for Common Methods

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 must 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 must 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 must 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 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, use any of the individual set methods to set the attribute value for an object for formats that are not natively supported; or write a format plug-in, set the format, and call the setProperties( ) method to invoke the new format plug-in. Otherwise, for formats that are 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.

## 2.4 Important Installation and Upgrade Considerations

A new database security measure introduced in Oracle Database 11*g* Release 2 (11.2) requires additional configuration steps for Oracle Multimedia applications using HTTP sources for media content. You can use the following query to determine if a media column contains HTTP sources. The query assumes that the table name is `MEDIA_TABLE` and the column name is `MEDIA_COLUMN`.

```
SELECT count(*)
 FROM MEDIA_TABLE m
 WHERE m.MEDIA_COLUMN.source.srcType = 'HTTP'
   AND m.MEDIA_COLUMN.source.local IS NOT NULL
   AND m.MEDIA_COLUMN.source.local <> 1
```

Oracle Multimedia uses the PL/SQL package UTL_HTTP to access media content for HTTP sources. Application users must have the appropriate permissions to connect to the remote host. For example, to grant the user `SCOTT` permission to access HTTP content located at the host `wwww.oracle.com:80`, the database administrator must execute the following commands:

```
SQL> REM Creates a new ACL and adds SCOTT the privilege to the ACL to make
SQL> REM TCP connections
SQL> EXECUTE DBMS_NETWORK_ACL_ADMIN.CREATE_ACL('acl_for_oracle.xml', -
> 'ACL for www.oracle.com', 'SCOTT', TRUE, 'connect')

SQL> REM Assigns the new ACL to www.oracle.com for TCP/IP port 80 (HTTP)
SQL> EXECUTE DBMS_NETWORK_ACL_ADMIN.ASSIGN_ACL('acl_for_oracle.xml', -
> 'www.oracle.com', 80)

SQL> REM Commits to make the ACL take effect
SQL> COMMIT
```

**See Also:**

- *Oracle Database PL/SQL Packages and Types Reference* for more information about configuring privileges for network access using the DBMS_NETWORK_ACL_ADMIN package
- *Oracle Database PL/SQL Packages and Types Reference* for more information about configuring privileges for network access using the UTL_HTTP package

# Common Methods

This section presents reference information about the Oracle Multimedia methods that are common to these Oracle Multimedia object types: ORDAudio, ORDDoc, ORDImage, and ORDVideo.

> **Note:** In this section, <*object-type*> represents exceptions that can be raised by ORDAudio, ORDDoc, ORDImage, or ORDVideo object types. Replace <*object-type*> with the appropriate object type.

Oracle Multimedia methods that are particular to an object type or implemented differently for each object type, are described in these chapters:

- ORDAudio Methods on page 3-8
- ORDDoc Methods on page 4-8
- ORDImage Methods on page 5-8
- ORDVideo Methods on page 6-8

This section describes these methods:

- clearLocal( ) on page 2-7
- closeSource( ) on page 2-8
- deleteContent( ) on page 2-10
- export( ) on page 2-11
- getBFile( ) on page 2-14
- getContent( ) on page 2-16
- getMimeType( ) on page 2-17
- getSource( ) on page 2-18
- getSourceLocation( ) on page 2-19
- getSourceName( ) on page 2-20
- getSourceType( ) on page 2-21
- getUpdateTime( ) on page 2-22
- isLocal( ) on page 2-23
- openSource( ) on page 2-24
- processSourceCommand( ) on page 2-26
- readFromSource( ) on page 2-28
- setLocal( ) on page 2-30
- setMimeType( ) on page 2-31
- setSource( ) on page 2-33
- setUpdateTime( ) on page 2-35
- trimSource( ) on page 2-36
- writeToSource( ) on page 2-38

**See Also:**

*Oracle Database Concepts* for more information about object types and methods

## 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

<*object-type*>Exceptions.NULL_SOURCE

This exception is raised when the value of the <*object-type*>.source attribute 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.

See Appendix G for more information about these exceptions.

### 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 parameter must be allocated and initialized to NULL. If you are using a user-defined source plug-in, call the openSource( ) method. (See Section 2.3.)

## 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 on.

## Pragmas

None.

## Exceptions

<*object-type*>Exceptions.NULL_SOURCE

This exception is raised when the value of the <*object-type*>.source attribute 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.

ORDSourceExceptions.INCOMPLETE_SOURCE_INFORMATION

This exception is raised if you call the closeSource( ) method and the value of 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.

See Appendix G 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

<*object-type*>Exceptions.NULL_SOURCE

This exception is raised when the value of the <*object-type*>.source attribute 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.

See Appendix G for more information about these exceptions.

## 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 provides native support only when the value of the source_type parameter is `FILE`. In this case, the data is written to a file within a directory that is accessible to Oracle Database. User-defined sources may support the export( ) method to provide WRITE access to other types of data stores.

### Parameters

**ctx**
The source plug-in context information. (See Section 2.3.)

**source_type**
The type of the external source data. This parameter is not case sensitive. (See Table 2–1.)

**source_location**
The location to which the source data is to be exported. (See Table 2–2.)

**source_name**
The name of the object to which the data is to be exported. (See Table 2–3.)

### 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 to delete the content of the source.localData attribute.

When the source_type parameter has a value of `FILE`, the source_location parameter specifies the name of an Oracle directory object, and the source_name parameter specifies the name of the file in which the data is to be contained.

The export( ) method writes only to a database directory object that the user has privilege to access. That is, you can access a directory object that you have created using the SQL statement CREATE DIRECTORY, or one to which you have been granted READ and WRITE access.

For example, the following SQL*Plus commands create a directory object and grant the user `ron` permission to read and write any file within the directory `c:\mydir\work`:

```
CONNECT sys as sysdba
```

```
Enter password: password
CREATE OR REPLACE DIRECTORY FILE_DIR AS 'c:\mydir\work';
GRANT READ,WRITE ON DIRECTORY FILE_DIR TO ron;
```

Now, the user `ron` can export an image to the `testimg.jpg` file in this directory using the export( ) method of the ORDImage object:

```
img.export('FILE', 'FILE_DIR', testimg.jpg');
```

See Section 2.1 for more information about directory and table definitions.

Invoking this method implicitly calls the setUpdateTime( ) method.

## Pragmas

None.

## Exceptions

<*object-type*>Exceptions.NULL_SOURCE

This exception is raised when the value of the <*object-type*>.source attribute 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.

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.IO_ERROR

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

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.

See Appendix G for more information about these exceptions.

## 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 read and write access to the directory for the user who will
-- be doing the exporting, in this case the user is ron.
CONNECT sys as sysdba
CREATE OR REPLACE DIRECTORY FILE_DIR as 'c:\mydir\work';
GRANT READ,WRITE ON DIRECTORY FILE_DIR TO 'ron';
BEGIN
-- Connect as the user ron:
CONNECT ron
Enter password: password
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 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

<*object-type*>Exceptions.NULL_SOURCE

This exception is raised when the value of the <*object-type*>.source attribute 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.

ORDSourceExceptions.INCOMPLETE_SOURCE_INFORMATION

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

ORDSourceExceptions.INVALID_SOURCE_TYPE

This exception is raised if the value of the source.srcType attribute is other than `FILE`.

See Appendix G for more information about these exceptions.

## 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

<*object-type*>Exceptions.NULL_SOURCE

This exception is raised when the value of the <*object-type*>.source attribute 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.

See Appendix G for more information about these exceptions.

## 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

<*object-type*>Exceptions.NULL_SOURCE

This exception is raised when the value of the <*object-type*>.source attribute 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.

See Appendix G for more information about these exceptions.

## 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

<*object-type*>Exceptions.NULL_SOURCE

This exception is raised when the value of the <*object-type*>.source attribute 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.

ORDSourceExceptions.INCOMPLETE_SOURCE_LOCATION

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

See Appendix G for more information about these exceptions.

### 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

<*object-type*>Exceptions.NULL_SOURCE

This exception is raised when the value of the <*object-type*>.source attribute 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.

ORDSourceExceptions.INCOMPLETE_SOURCE_NAME

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

See Appendix G for more information about these exceptions.

## 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

<*object-type*>Exceptions.NULL_SOURCE

This exception is raised when the value of the <*object-type*>.source attribute 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.

See Appendix G for more information about these exceptions.

### 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

<*object-type*>Exceptions.NULL_SOURCE

This exception is raised when the value of the <*object-type*>.source attribute 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.

See Appendix G for more information about these exceptions.

## 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

<*object-type*>Exceptions.NULL_SOURCE

This exception is raised when the value of the <*object-type*>.source attribute 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.

See Appendix G for more information about these exceptions.

### Examples

Determine whether 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 parameter can be used by user-defined source plug-ins.

**ctx**
The source plug-in context information. (See Section 2.3.)

## 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 on.

## Pragmas

None.

## Exceptions

<*object-type*>Exceptions.NULL_SOURCE

This exception is raised when the value of the <*object-type*>.source attribute 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.

ORDSourceExceptions.INCOMPLETE_SOURCE_INFORMATION

This exception is raised if you call the openSource( ) method and the value of 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.

See Appendix G 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 parameter must be allocated and initialized to NULL. If you are using a user-defined source plug-in, call the openSource( ) method. (See Section 2.3.)

**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

<*object-type*>Exceptions.NULL_SOURCE

This exception is raised when the value of the <*object-type*>.source attribute 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.

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.

See Appendix G 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 parameter must be allocated and initialized to NULL. If you are using a user-defined source plug-in, call the openSource( ) method. (See Section 2.3.)

**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 is to 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. 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

<*object-type*>Exceptions.NULL_SOURCE

This exception is raised when the value of the <*object-type*>.source attribute 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.

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.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.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`.

See Appendix G 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 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 value of the source.localData attribute 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.

<*object-type*>Exceptions.NULL_SOURCE

This exception is raised when the value of the <*object-type*>.source attribute 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.

See Appendix G for more information about these exceptions.

## 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 of 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.

<*object-type*>Exceptions.NULL_SOURCE

This exception is raised when the value of the <*object-type*>.source attribute 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.

See Appendix G for more information about these exceptions.

### 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 Table 2–1.)

**source_location**
The location of the external source data. (See Table 2–2.)

**source_name**
The name of the external source data. (See Table 2–3.)

### 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

<*object-type*>Exceptions.NULL_SOURCE

This exception is raised when the value of the <*object-type*>.source attribute 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.

ORDSourceExceptions.INCOMPLETE_SOURCE_INFORMATION

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

See Appendix G for more information about these exceptions.

### 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 time stamp 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

<*object-type*>Exceptions.NULL_SOURCE

This exception is raised when the value of the <*object-type*>.source attribute 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.

See Appendix G for more information about these exceptions.

### 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 parameter must be allocated and initialized to NULL. If you are using a user-defined source plug-in, call the openSource( ) method. (See Section 2.3.)

**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 on.

## Pragmas

None.

## Exceptions

<*object-type*>Exceptions.NULL_SOURCE

This exception is raised when the value of the <*object-type*>.source attribute 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.

ORDSourceExceptions.INCOMPLETE_SOURCE_INFORMATION

This exception is raised if you call the trimSource( ) method and the value of 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.

See Appendix G 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 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 parameter must be allocated and initialized to NULL. If you are using a user-defined source plug-in, call the openSource( ) method. (See Section 2.3.)

**startPos**
The start position in the source to where the buffer is to 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 do not permit you to write, and do not support this method. This method works if data is stored in a local BLOB or is accessible through a user-defined source plug-in.

## Pragmas

None.

## Exceptions

<*object-type*>Exceptions.NULL_SOURCE

This exception is raised when the value of the <*object-type*>.source attribute 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.

ORDSourceExceptions.INCOMPLETE_SOURCE_INFORMATION

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

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.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`.

See Appendix G 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;
/
```

# 3

# Oracle Multimedia ORDAudio Object Type

Oracle Multimedia provides the ORDAudio object type, which supports the storage and management of audio data.

The ORDAudio object type is defined in the `ordaspec.sql` file. After installation, this file is available in the Oracle home directory at:

*<ORACLE_HOME>*`/ord/im/admin` (on Linux and UNIX)

*<ORACLE_HOME>*`\ord\im\admin` (on Windows)

This chapter includes these sections:



## 3.1 ORDAudio Object Examples

The examples in this chapter use the ONLINE_MEDIA table in the Product Media (PM) sample schema. To replicate the examples on your computer, begin with the examples shown in the reference pages for the ORDAudio constructors and the import( ) and importFrom( ) methods. Then, substitute your audio files for those in the examples.

The example for the constructor ORDAudio for BLOBs uses the test audio table TAUD (see TAUD Table Definition).

---

**Note:** If you manipulate the audio data itself (by either directly modifying the BLOB or changing the external source), 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.

---

**See Also:**

*Oracle Database Sample Schemas* for more information about the PM schema

### 3.1.1 Directory Definition for ORDAudio Object Examples

Issue the following statements before executing the examples, where `c:\mydir\work` is the directory where the user `ron` can find the audio data:

```
CONNECT sys as sysdba
CREATE OR REPLACE DIRECTORY FILE_DIR as 'c:\mydir\work';
GRANT READ ON DIRECTORY FILE_DIR TO 'ron';
```

## 3.2 Important Notes for ORDAudio Methods

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 must 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 must 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 must 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 use the ctx argument, but if you code as previously described, your application should work with current or future source or format plug-ins.

---

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. The attributes for this object type are defined as follows in the `ordaspec.sql` file:

```
--------------------
-- TYPE 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,
```

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. Oracle recommends that you not write to this attribute directly.

## ORDAudio Constructors

This section describes these ORDAudio constructor functions:

## ORDAudio for BLOBs

### Format

```
ORDAudio(SELF IN OUT NOCOPY ORDSYS.ORDAudio,
         data IN BLOB,
         setproperties IN INTEGER DEFAULT 0)
RETURN SELF AS RESULT
```

### Description

Constructs an ORDAudio object from a BLOB. The data stored in the BLOB is copied into the ORDAudio object when the constructed ORDAudio object is inserted or updated into a table.

### Parameters

**data**
Audio content stored in a BLOB.

**setproperties**
Indicator flag that determines whether the setProperties( ) method is called in the constructor. If the value is `1`, the setProperties( ) method is called. If the value is `0`, the method is not called. The default is `0`.

### Pragmas

None.

### Exceptions

None.

### Usage Notes

Use this constructor to create an ORDAudio object when the audio content is stored in either a temporary or a persistent BLOB.

### Examples

Create an ORDAudio object from a BLOB object and insert it into the table.

> **Note:** The user who runs this statement must have the SELECT privilege on the table `taud` created by the user `mediauser`.
>
> See Section 7.2.2 for the definition of the test audio table TAUD.

```
INSERT INTO pm.online_media (product_id, product_audio)
  SELECT 1729, ORDSYS.ORDAudio(t.aud) FROM mediauser.taud t WHERE t.n = 1;
```

# ORDAudio for Other Sources

## Format

```
ORDAudio( SELF IN OUT NOCOPY ORDSYS.ORDAudio,
                  source_type      IN VARCHAR2 DEFAULT 'LOCAL',
                  source_location  IN VARCHAR2 DEFAULT NULL,
                  source_name      IN VARCHAR2 DEFAULT NULL,
                  setproperties  IN INTEGER DEFAULT 0
) RETURN SELF AS RESULT
```

## Description

Constructs an ORDAudio object from a specific source. By default, an empty object with a local source is constructed.

## Parameters

**source_type**
The type of the source audio data. Valid values are: `FILE`, `HTTP`, `LOCAL`, or user-defined. The default is `LOCAL`.

The parameter value `LOCAL` indicates that the data is stored in Oracle Database. The parameter value `LOCAL` is never stored in the srcType attribute. Rather, this parameter value indicates that the data is stored in Oracle Database in the localData attribute. (See Section 2.2 for a description of the ORDSource object.)

**source_location**
The location from which the source audio data is to be imported. (See Table 2–2.)

**source_name**
The name of the source audio data. (See Table 2–3.)

**setproperties**
Indicator flag that determines whether the setProperties( ) method is called in the constructor. If the value is `1`, the setProperties( ) method is called. If the value is `0`, the method is not called. The default is `0`.

## Pragmas

None.

## Exceptions

None.

## Usage Notes

Use this constructor to create an ORDAudio object when the audio content is not already stored in a BLOB, as in any of these situations:

The content is in a local file.
The content is referenced by a URL.
The content is not available when the ORDAudio object is created.

## Examples

Create an ORDAudio object from a specified source:

```
-- Construct an ORDAudio object from an external file and insert it into the
table.
INSERT INTO pm.online_media (product_id, product_audio)
  VALUES (1733, ORDSYS.ORDAudio('FILE', 'FILE_DIR', 'speaker.au'));

-- Initialize an ORDAudio object in the table.
INSERT INTO pm.online_media (product_id, product_audio)
  VALUES(1729, ORDSYS.ORDAudio());
```

# ORDAudio Methods

This section presents reference information about the Oracle Multimedia methods used specifically for audio data manipulation.

Chapter 2 presents reference information about the Oracle Multimedia 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.

This section describes these methods:



**See Also:**

*Oracle Database Concepts* for more information about object types and methods

## checkProperties( )

### Format

checkProperties(ctx IN OUT RAW) RETURN BOOLEAN;

### Description

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

### Parameters

**ctx**
The format plug-in context information. (See Section 3.2.)

### 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.

ORDAudioExceptions.NULL_SOURCE

This exception is raised when the value of the ORDAudio.source attribute is NULL.

See Appendix G for more information about these exceptions.

### 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 this 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. (See Section 3.2.)

**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.

ORDAudioExceptions.NULL_SOURCE

This exception is raised when the value of the ORDAudio.source attribute is NULL.

See Appendix G for more information about these exceptions.

### 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. (See Section 3.2.)

**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 Also:**
>
> *Oracle Multimedia User's Guide* for more information about extending support for media formats

### 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.

ORDAudioExceptions.NULL_SOURCE

This exception is raised when the value of the ORDAudio.source attribute is `NULL`.

See Appendix G for more information about these exceptions.

### 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 is 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 3-39.

# 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 3-40.

## 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. (See Section 3.2.)

### Usage Notes

None.

### Pragmas

None.

### Exceptions

ORDAudioExceptions.NULL_SOURCE

This exception is raised when the value of the ORDAudio.source attribute is `NULL`.

ORDSourceExceptions.INCOMPLETE_SOURCE_INFORMATION

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

See Appendix G for more information about these exceptions.

### Examples

See the example in import( ) on page 3-27.

# 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. (See Section 3.2.)

**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

ORDAudioExceptions.NULL_SOURCE

This exception is raised when the value of the ORDAudio.source attribute is NULL.

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.

See Appendix G 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.

See Appendix G for more information about this exception.

## 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 3-42.

# 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 of the format attribute is `NULL`.

See Appendix G for more information about this exception.

## Examples

See the example in setProperties( ) on page 3-42.

## 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 3-42.

# 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 3-42.

## 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 3-42.

# 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) within the database.

## Parameters

**ctx**
The source plug-in context information. This parameter must be allocated and initialized to NULL. If you are using a user-defined source plug-in, call the openSource( ) method. (See Section 3.2.)

## 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 before calling the import( ) method.

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.

If the value of the source.srcType attribute is `FILE`, the source.srcLocation attribute contains the name of a database directory object that contains the file to be imported, and the source.srcName attribute contains the name of the file to be imported. You must ensure that the directory for the external source location exists or is created before you use this method.

The import( ) method reads only from a database directory object that the user has privilege to access. That is, you can access a directory object that you have created using the SQL statement CREATE DIRECTORY, or one to which you have been granted READ access.

For example, the following SQL*Plus commands create a directory object and grant the user `ron` permission to read any file within the directory `c:\mydir\work`:

```
CONNECT sys as sysdba
Enter password: password
CREATE OR REPLACE DIRECTORY FILE_DIR AS 'c:\mydir\work';
GRANT READ ON DIRECTORY FILE_DIR TO ron;
```

See Section 3.1 for more information about directory and table definitions.

If the value of the source.srcType attribute is `HTTP`, the source.srcLocation attribute contains the base URL needed to find the directory that contains the object to be imported, and the source.srcName attribute contains the name of the object to be imported.

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 Linux and UNIX, setting the environment variable http_proxy to a URL specifies that the UTL_HTTP package must 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 not be used for URLs in the specified domain.

**See Also:**

*Oracle Database PL/SQL Packages and Types Reference* for more information about the UTL_HTTP package

If the value of the source.srcType attribute is a user-defined name, the source.srcLocation attribute contains an identifier string required to access the user-defined object to be imported, and the source.srcName attribute contains the name of the object to be imported.

## Pragmas

None.

## Exceptions

ORDAudioExceptions.NULL_SOURCE

This exception is raised when the value of the ORDAudio.source attribute is NULL.

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 the import( ) method is not supported by the source plug-in being used.

ORDSourceExceptions.NULL_SOURCE

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

See Appendix G 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) within the database.

## Parameters

**ctx**
The source plug-in context information. This parameter must be allocated and initialized to NULL. If you are using a user-defined source plug-in, call the openSource( ) method. (See Section 3.2.)

**source_type**
The type of the source audio data. (See Table 2–1.)

**source_location**
The location from which the source audio data is to be imported. (See Table 2–2.)

**source_name**
The name of the source audio data. (See Table 2–3.)

## Usage Notes

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

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.

If the value of the source_type parameter is `FILE`, the source_location parameter contains the name of a database directory object that contains the file to be imported, and the source_name parameter contains the name of the file to be imported. You must ensure that the directory indicated by the source_location parameter exists or is created before you use this method.

The importFrom( ) method reads only from a database directory object that the user has privilege to access. That is, you can access a directory object that you have created using the SQL statement CREATE DIRECTORY, or one to which you have been granted READ access.

For example, the following SQL*Plus commands create a directory object and grant the user `ron` permission to read any file within the directory `c:\mydir\work`:

```
CONNECT sys as sysdba
Enter password: password
CREATE OR REPLACE DIRECTORY FILE_DIR AS 'c:\mydir\work';
```

```
GRANT READ ON DIRECTORY FILE_DIR TO ron;
```

See Section 3.1 for more information about directory and table definitions.

If the value of the source_type parameter is HTTP, the source_location parameter contains the base URL needed to find the directory that contains the object to be imported, and the source_name parameter contains the name of the object to be imported.

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 Linux and UNIX, setting the environment variable http_proxy to a URL specifies that the UTL_HTTP package must 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 not be used for URLs in the specified domain.

**See Also:**

*Oracle Database PL/SQL Packages and Types Reference* for more information about the UTL_HTTP package

If the value of the source_type parameter is a user-defined name, the source_location parameter contains an identifier string required to access the user-defined object to be imported, and the source_name parameter contains the name of the object to be imported.

## Pragmas

None.

## Exceptions

ORDAudioExceptions.NULL_SOURCE

This exception is raised when the value of the ORDAudio.source 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.NULL_SOURCE

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

See Appendix G 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. (See Section 3.2.)

**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 Also:**
>
> *Oracle Multimedia User's Guide* for more information about extending support for media formats

### 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.

ORDAudioExceptions.NULL_SOURCE

This exception is raised when the value of the ORDAudio.source attribute is `NULL`.

See Appendix G for more information about these exceptions.

## 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 of the knownAudioDuration parameter is `NULL`.

ORDAudioExceptions.NULL_SOURCE

This exception is raised when the value of the ORDAudio.source attribute is `NULL`.

See Appendix G for more information about these exceptions.

### Examples

See the example in setFormat( ) on page 3-37.

# 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 of the knownCompressionType parameter is `NULL`.

ORDAudioExceptions.NULL_SOURCE

This exception is raised when the value of the ORDAudio.source attribute is `NULL`.

See Appendix G for more information about these exceptions.

## Examples

See the example in setFormat( ) on page 3-37.

## 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 Oracle Multimedia make use of this description attribute to present audio data to users.

Calling this method implicitly calls the setUpdateTime( ) method.

### Pragmas

None.

### Exceptions

ORDAudioExceptions.NULL_SOURCE

This exception is raised when the value of the ORDAudio.source attribute is NULL.

See Appendix G for more information about this exception.

### 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 of the knownEncoding parameter is `NULL`.

ORDAudioExceptions.NULL_SOURCE

This exception is raised when the value of the ORDAudio.source attribute is `NULL`.

See Appendix G for more information about these exceptions.

## Examples

See the example in setFormat( ) on page 3-37.

## 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 of the knownFormat parameter is NULL.

ORDAudioExceptions.NULL_SOURCE

This exception is raised when the value of the ORDAudio.source attribute is NULL.

See Appendix G for more information about these exceptions.

### 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

ORDAudioExceptions.NULL_SOURCE

This exception is raised when the value of the ORDAudio.source attribute is `NULL`.

See Appendix G for more information about this exception.

## 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 of the knownNumberOfChannels parameter is `NULL`.

ORDAudioExceptions.NULL_SOURCE

This exception is raised when the value of the ORDAudio.source attribute is `NULL`.

See Appendix G for more information about these exceptions.

### Examples

See the example in setFormat( ) on page 3-37.

# 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 these 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. (See Section 3.2.)

**setComments**
A Boolean value that indicates whether 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.

ORDAudioExceptions.NULL_SOURCE

This exception is raised when the value of the ORDAudio.source attribute is `NULL`.

See Appendix G for more information about these exceptions.

## 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 = 1729;
 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 of the knownSamplingRate parameter is `NULL`.

ORDAudioExceptions.NULL_SOURCE

This exception is raised when the value of the ORDAudio.source attribute is `NULL`.

See Appendix G for more information about these exceptions.

## Examples

See the example in setFormat( ) on page 3-37.

## 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 of the knownSampleSize parameter is `NULL`.

ORDAudioExceptions.NULL_SOURCE

This exception is raised when the value of the ORDAudio.source attribute is `NULL`.

See Appendix G for more information about these exceptions.

### Examples

See the example in setFormat( ) on page 3-37.

# 4

# Oracle Multimedia ORDDoc Object Type

Oracle Multimedia provides the ORDDoc object type, which supports the storage and management of any media data including image, audio, and video.

The ORDDoc object type is defined in the `orddspec.sql` file. After installation, this file is available in the Oracle home directory at:

*<ORACLE_HOME>*`\ord\im\admin` (on Windows)

*<ORACLE_HOME>*`/ord/im/admin` (on Linux and UNIX)

This chapter contains these sections:



## 4.1 ORDDoc Object Examples

The examples in this chapter use the ONLINE_MEDIA table in the Product Media (PM) sample schema. To replicate the examples on your computer, begin with the examples shown in the reference pages for the ORDDoc constructors and the import( ) and importFrom( ) methods. Then, substitute your media files for those in the examples.

The example for the constructor ORDDoc for BLOBs uses the test document table TDOC (see TDOC Table Definition).

---

**Note:** If you manipulate the media data itself (by either directly modifying the BLOB or changing the external source), 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.

---

**See Also:**

*Oracle Database Sample Schemas* for more information about the PM schema

### 4.1.1 Directory Definition for ORDDoc Object Examples

Issue the following statements before executing the examples, where `c:\mydir\work` is the directory where the user `ron` can find the image, audio, and video data:

```
CONNECT sys as sysdba
CREATE OR REPLACE DIRECTORY FILE_DIR as 'c:\mydir\work';
GRANT READ ON DIRECTORY FILE_DIR TO 'ron';
```

## 4.2 Important Notes for ORDDoc Methods

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 must 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 must 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 must 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 use the ctx argument, but if you code as previously described, your application should work with current or future source or format plug-ins.

---

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. The attributes for this object type are defined as follows in the `orddspec.sql` file:

```
------------------
-- TYPE ATTRIBUTES
------------------
source              ORDSource,
format              VARCHAR(80),
mimeType            VARCHAR(80),
contentLength       INTEGER,
comments            CLOB,
```

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. Oracle recommends that you not write to this attribute directly.

## ORDDoc Constructors

This section describes these ORDDoc constructor functions:

## ORDDoc for BLOBs

### Format

```
ORDDoc(SELF IN OUT NOCOPY ORDSYS.ORDDoc,
       data IN BLOB,
       setproperties IN INTEGER DEFAULT 0)
RETURN SELF AS RESULT
```

### Description

Constructs an ORDDoc object from a BLOB. The data stored in the BLOB is copied into the ORDDoc object when the constructed ORDDoc object is inserted or updated into a table.

### Parameters

**data**
Media content stored in a BLOB.

**setproperties**
Indicator flag that determines whether the setProperties( ) method is called in the constructor. If the value is `1`, the setProperties( ) method is called. If the value is `0`, the method is not called. The default is `0`.

### Pragmas

None.

### Exceptions

None.

### Usage Notes

Use this constructor to create an ORDDoc object when the media content is stored in either a temporary or a persistent BLOB.

### Examples

Create an ORDDoc object from a BLOB object and insert it into the table.

> **Note:** The user who runs this statement must have the SELECT privilege on the table `tdoc` created by the user `mediauser`.
>
> See Section 7.2.3 for a definition of the test document table TDOC.

```
INSERT INTO pm.online_media (product_id, product_testimonials)
  SELECT 2808, ORDSYS.ORDDoc(t.document) FROM mediauser.tdoc t WHERE t.n = 1;
```

# ORDDoc for Other Sources

## Format

```
ORDDoc( SELF IN OUT NOCOPY ORDSYS.ORDDoc,
                source_type      IN VARCHAR2 DEFAULT 'LOCAL',
                source_location  IN VARCHAR2 DEFAULT NULL,
                source_name      IN VARCHAR2 DEFAULT NULL,
                setproperties  IN INTEGER DEFAULT 0
) RETURN SELF AS RESULT
```

## Description

Constructs an ORDDoc object from a specific source. By default, an empty object with a local source is constructed.

## Parameters

**source_type**
The type of the source media data. Valid values are: `FILE`, `HTTP`, `LOCAL`, or user-defined. The default is `LOCAL`.

The parameter value `LOCAL` indicates that the data is stored in Oracle Database. The parameter value `LOCAL` is never stored in the srcType attribute. Rather, this parameter value indicates that the data is stored in Oracle Database in the localData attribute. (See Section 2.2 for a description of the ORDSource object.)

**source_location**
The location from which the source media data is to be imported. (See Table 2–2.)

**source_name**
The name of the source media data. (See Table 2–3.)

**setproperties**
Indicator flag that determines whether the setProperties( ) method is called in the constructor. If the value is `1`, the setProperties( ) method is called. If the value is `0`, the method is not called. The default is `0`.

## Pragmas

None.

## Exceptions

None.

## Usage Notes

Use this constructor to create an ORDDoc object when the media content is not already stored in a BLOB, as in any of these situations:

The content is in a local file.
The content is referenced by a URL.
The content is not available when the ORDDoc object is created.

## Examples

Create an ORDDoc object from a specified source:

```
-- Construct an ORDDoc object from an external file and insert it into the table.
INSERT INTO pm.online_media (product_id, product_testimonials)
  VALUES (2999, ORDSYS.ORDDoc('FILE', 'FILE_DIR', 'modem.jpg'));

-- Initialize an ORDDoc object in the table.
INSERT INTO pm.online_media (product_id, product_testimonials)
  VALUES(2808, ORDSYS.ORDDoc());
```

# ORDDoc Methods

This section presents reference information about the Oracle Multimedia methods used specifically for media data manipulation.

Chapter 2 presents reference information about the Oracle Multimedia 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.

This section describes these methods:

- getContentInLob( ) on page 4-9
- getContentLength( ) on page 4-11
- getFormat( ) on page 4-12
- import( ) on page 4-13
- importFrom( ) on page 4-16
- setFormat( ) on page 4-19
- setProperties( ) on page 4-20

**See Also:**

*Oracle Database Concepts* for more information about object types and methods

## 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. (See Section 4.2.)

**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

ORDDocExceptions.NULL_SOURCE

This exception is raised when the value of the ORDDoc.source attribute is `NULL`.

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.

See Appendix G 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

None.

### Exceptions

ORDDocExceptions.NULL_SOURCE

This exception is raised when the value of the ORDDoc.source attribute is `NULL`.

See Appendix G for more information about this exception.

### Examples

Get the content length of the media data:

```
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`.

See Appendix G for more information about this exception.

## Examples

See the example in setFormat( ) on page 4-19.

# 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 parameter must be allocated and initialized to NULL. If you are using a user-defined source plug-in, call the openSource( ) method. (See Section 4.2.)

**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 setProperties( ) method is not called. The default value is `FALSE`.

## 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 before 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.

If the value of the source.srcType attribute is `FILE`, the source.srcLocation attribute contains the name of a database directory object that contains the file to be imported, and the source.srcName attribute contains the name of the file to be imported. You must ensure that the directory for the external source location exists or is created before you use this method.

The import( ) method reads only from a database directory object that the user has privilege to access. That is, you can access a directory object that you have created using the SQL statement CREATE DIRECTORY, or one to which you have been granted READ access.

For example, the following SQL*Plus commands create a directory object and grant the user `ron` permission to read any file within the directory `c:\mydir\work`:

```
CONNECT sys as sysdba
Enter password: password
CREATE OR REPLACE DIRECTORY FILE_DIR AS 'c:\mydir\work';
GRANT READ ON DIRECTORY FILE_DIR TO ron;
```

See Section 4.1 for more information about directory and table definitions.

If the value of the source.srcType attribute is HTTP, the source.srcLocation attribute contains the base URL needed to find the directory that contains the object to be imported, and the source.srcName attribute contains the name of the object to be imported.

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 Linux and UNIX, setting the environment variable http_proxy to a URL specifies that the UTL_HTTP package must 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 not be used for URLs in the specified domain.

**See Also:**

*Oracle Database PL/SQL Packages and Types Reference* for more information about the UTL_HTTP package

If the value of the source.srcType attribute is a user-defined name, the source.srcLocation attribute contains an identifier string required to access the user-defined object to be imported, and the source.srcName attribute contains the name of the object to be imported.

## Pragmas

None.

## Exceptions

ORDDocExceptions.NULL_SOURCE

This exception is raised when the value of the ORDDoc.source attribute is NULL.

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 the import( ) method is not supported by the source plug-in being used.

ORDSourceExceptions.NULL_SOURCE

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

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 G 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 parameter must be allocated and initialized to NULL. If you are using a user-defined source plug-in, call the openSource( ) method. (See Section 4.2.)

**source_type**
The type of the source media data. (See Table 2–1.)

**source_location**
The location from which the source media data is to be imported. (See Table 2–2.)

**source_name**
The name of the source media data. (See Table 2–3.)

**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 setProperties( ) 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.

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.

If the value of the source_type parameter is `FILE`, the source_location parameter contains the name of a database directory object that contains the file to be imported, and the source_name parameter contains the name of the file to be imported. You must ensure that the directory indicated by the source_location parameter exists or is created before you use this method.

The importFrom( ) method reads only from a database directory object that the user has privilege to access. That is, you can access a directory object that you have created using the SQL statement CREATE DIRECTORY, or one to which you have been granted READ access.

For example, the following SQL*Plus commands create a directory object and grant the user `ron` permission to read any file within the directory `c:\mydir\work`:

```
CONNECT sys as sysdba
Enter password: password
CREATE OR REPLACE DIRECTORY FILE_DIR AS 'c:\mydir\work';
GRANT READ ON DIRECTORY FILE_DIR TO ron;
```

See Section 4.1 for more information about directory and table definitions.

If the value of the source_type parameter is `HTTP`, the source_location parameter contains the base URL needed to find the directory that contains the object to be imported, and the source_name parameter contains the name of the object to be imported.

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 Linux and UNIX, setting the environment variable http_proxy to a URL specifies that the UTL_HTTP package must 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 not be used for URLs in the specified domain.

**See Also:**

*Oracle Database PL/SQL Packages and Types Reference* for more information about the UTL_HTTP package

If the value of the source_type parameter is a user-defined name, the source_location parameter contains an identifier string required to access the user-defined object to be imported, and the source_name parameter contains the name of the object to be imported.

## Pragmas

None.

## Exceptions

ORDDocExceptions.NULL_SOURCE

This exception is raised when the value of the ORDDoc.source attribute is `NULL`.

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.

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 G 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 of the knownFormat parameter is NULL.

ORDDocExceptions.NULL_SOURCE

This exception is raised when the value of the ORDDoc.source attribute is NULL.

See Appendix G for more information about these exceptions.

### 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 these 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, respectively, for information about natively supported audio, image, and video formats.

## Parameters

**ctx**
The format plug-in context information. (See Section 4.2.)

**setComments**
A Boolean value that indicates whether 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.

ORDDocExceptions.NULL_SOURCE

This exception is raised when the value of the ORDDoc.source attribute is `NULL`.

See Appendix G for more information about these exceptions.

## Examples

**Example 1:**

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;
/
```

**Example 2:**

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;
/
```

# 5

# Oracle Multimedia ORDImage Object Type

Oracle Multimedia provides the ORDImage object type, which supports the storage, management, and manipulation of image data.

The ORDImage object type is defined in the `ordispec.sql` file. After installation, this file is available in the Oracle home directory at:

*<ORACLE_HOME>*`/ord/im/admin` (on Linux and UNIX)

*<ORACLE_HOME>*`\ord\im\admin` (on Windows)

This chapter contains the following sections:



## 5.1 ORDImage Object Examples

The examples in this chapter use the ONLINE_MEDIA table in the Product Media (PM) sample schema. To replicate the examples on your computer, begin with the examples shown in the reference pages for the ORDImage constructors and the import( ) and importFrom( ) methods. Then, substitute your image files for those in the examples.

The example for the constructor ORDImage for BLOBs uses the test image table TIMG (see TIMG Table Definition).

---

**Note:** If you manipulate the image data itself (by either directly modifying the BLOB or changing the external source), 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.

---

**See Also:**

*Oracle Database Sample Schemas* for more information about the PM schema

### 5.1.1 Directory Definition for ORDImage Object Examples

Issue the following statements before executing the examples, where `c:\mydir\work` is the directory where the user `ron` can find the image data:

```
CONNECT sys as sysdba
CREATE OR REPLACE DIRECTORY FILE_DIR as 'c:\mydir\work';
GRANT READ ON DIRECTORY FILE_DIR TO 'ron';
```

## 5.2 Important Notes for ORDImage Methods

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 must 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 must 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 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 formats that are natively supported, use the setProperties( ) method to automatically populate the attributes of the object. Otherwise, use the setProperties( ) method for foreign images; or use any of the individual set methods to set the attribute value for an object for formats that are not natively supported.

# ORDImage Object Type

The ORDImage object type supports the storage, management, and manipulation of image data. The attributes for this object type are defined as follows in the `ordispec.sql` file:

```
------------------
-- TYPE ATTRIBUTES
------------------
source              ORDSource,
height              INTEGER,
width               INTEGER,
contentLength       INTEGER,
fileFormat          VARCHAR2(4000),
contentFormat       VARCHAR2(4000),
compressionFormat   VARCHAR2(4000),
mimeType            VARCHAR2(4000),
```

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, JFIF, and so on).
- contentFormat: the type of image (monochrome and so on).
- compressionFormat: the compression algorithm used on the image data.
- mimeType: the MIME type information.

## ORDImage Constructors

This section describes these ORDImage constructor functions:

## ORDImage for BLOBs

### Format

```
ORDImage(SELF IN OUT NOCOPY ORDSYS.ORDImage,
                data IN BLOB,
                setproperties IN INTEGER DEFAULT 0)
RETURN SELF AS RESULT
```

### Description

Constructs an ORDImage object from a BLOB. The data stored in the BLOB is copied into the ORDImage object when the constructed ORDImage object is inserted or updated into a table.

### Parameters

**data**
Image content stored in a BLOB.

**setproperties**
Indicator flag that determines whether the setProperties( ) method is called in the constructor. If the value is `1`, the setProperties( ) method is called. If the value is `0`, the method is not called. The default is `0`.

### Pragmas

None.

### Exceptions

None.

### Usage Notes

Use this constructor to create an ORDImage object when the image content is stored in either a temporary or a persistent BLOB.

### Examples

Create an ORDImage object from a BLOB object and insert it into the table.

> **Note:** The user who runs this statement must have the SELECT privilege on the table `timg` created by the user `mediauser`.
>
> See Section 7.2.4 for a definition of the test image table TIMG.

```
INSERT INTO pm.online_media (product_id, product_photo)
  SELECT 3501, ORDSYS.ORDImage(t.img) FROM mediauser.timg t WHERE t.n = 1;
```

# ORDImage for Other Sources

## Format

```
ORDImage( SELF IN OUT NOCOPY ORDSYS.ORDImage,
                  source_type      IN VARCHAR2 DEFAULT 'LOCAL',
                  source_location  IN VARCHAR2 DEFAULT NULL,
                  source_name      IN VARCHAR2 DEFAULT NULL,
                  setproperties  IN INTEGER DEFAULT 0
) RETURN SELF AS RESULT
```

## Description

Constructs an ORDImage object from a specific source. By default, an empty object with a local source is constructed.

## Parameters

**source_type**
The type of the source image data. Valid values are: `FILE`, `HTTP`, `LOCAL`, or user-defined. The default is `LOCAL`.

The parameter value `LOCAL` indicates that the data is stored in Oracle Database. The parameter value `LOCAL` is never stored in the srcType attribute. Rather, this parameter value indicates that the data is stored in Oracle Database in the localData attribute. (See Section 2.2 for a description of the ORDSource object.)

**source_location**
The location from which the source image data is to be imported. (See Table 2–2.)

**source_name**
The name of the source image data. (See Table 2–3.)

**setproperties**
Indicator flag that determines whether the setProperties( ) method is called in the constructor. If the value is `1`, the setProperties( ) method is called. If the value is `0`, the method is not called. The default is `0`.

## Pragmas

None.

## Exceptions

None.

## Usage Notes

Use this constructor to create an ORDImage object when the image content is not stored in a BLOB, as in any of these situations:

The content is in a local file.
The content is referenced by a URL.
The content is not available when the ORDImage object is created.

## Examples

Create an ORDImage object from a specified source:

```
-- Construct an ORDImage object from an external file and insert it into the
table.
INSERT INTO pm.online_media (product_id, product_photo)
  VALUES (3515, ORDSYS.ORDImage('FILE', 'FILE_DIR', 'speaker.jpg'));

-- Initialize an ORDImage object in the table.
INSERT INTO pm.online_media (product_id, product_photo)
  VALUES(3501, ORDSYS.ORDImage());
```

# ORDImage Methods

This section presents reference information about the Oracle Multimedia methods used specifically for image data manipulation.

Chapter 2 presents reference information about the Oracle Multimedia 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.

This section describes these methods:



**See Also:**

*Oracle Database Concepts* for more information about object types and methods

## applyWatermark( ) image

### Format

```
applyWatermark(SELF     IN OUT NOCOPY ORDImage,
               added_image IN OUT NOCOPY ORDImage,
               dest     IN OUT NOCOPY ORDImage,
               logging OUT VARCHAR2,
               watermark_properties IN ORDSYS.ORD_STR_LIST default null),
```

### Description

Overlays an image watermark onto a source image and writes it to a destination image object.

### Parameters

**added_image**
The watermark image stored in an ORDImage object to be added to the source image.

**dest**
The destination ORDImage object for the watermarked image.

**logging**
A string that contains information about any unexpected behavior that occurred during the watermarking operation. If the operation is successful, an empty string is returned.

**watermark_properties**
A string list of name-value pairs that define the attributes of the watermark image to be overlaid, including: width, height, position, position_x, position_y, and transparency. See Table D–2 for the complete list of watermark properties.

### Usage Notes

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).

See Section D.6 for more information about watermarking operations and watermark properties.

### Pragmas

None.

### Exceptions

ORDImageExceptions.DATA_NOT_LOCAL

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

ORDImageExceptions.NULL_DESTINATION

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

ORDImageExceptions.NULL_LOCAL_DATA

This exception is raised if you call the applyWatermark( ) method and the value of the destination image source.localData attribute is NULL. This exception is also raised if you call the applyWatermark( ) method and the value of the source image source.local attribute is 1 or NULL, and the value of the source.localData attribute is NULL.

ORDImageExceptions.NULL_SOURCE

This exception is raised if the source or watermark image source attribute value is NULL.

See Appendix G for more information about these exceptions.

## Examples

Add a watermark image to a source image:

```
-- add image as watermark to source image
DECLARE
  source_image ORDSYS.ORDImage;
  added_image ORDSYS.ORDImage;
  dest_image ORDSYS.ORDImage;
  prop ordsys.ord_str_list;
  logging VARCHAR2(2000);
BEGIN
  select product_photo into source_image from pm.online_media
    where product_id = 4001;
  select product_photo into added_image from pm.online_media
    where product_id = 4002;
  select product_photo into dest_image from pm.online_media
    where product_id = 4003 for update;

  -- specify properties
  prop := ordsys.ord_str_list(
                  'position=bottomright',
                  'transparency=0.2');

  -- add image watermark to source image
  source_image.applyWatermark(added_image, dest_image, logging, prop);
  update pm.online_media set product_photo = dest_image where product_id = 4003;
  commit;
EXCEPTION
  WHEN OTHERS THEN
      RAISE;
END;
/
```

where:

- product_id 4001: the source image (in JPEG format) stored in the table. The source image is larger than the watermark image.
- product_id 4002: the watermark image (in PNG format) stored in the table. The watermark image is overlaid onto the bottom right position of the source image.
- product_id 4003: the resulting watermarked image stored in the table.

## applyWatermark( ) text

### Format

```
applyWatermark(SELF    IN OUT NOCOPY ORDImage,
               added_text IN    VARCHAR2,
               dest    IN OUT NOCOPY ORDImage,
               logging OUT VARCHAR2,
               watermark_properties IN ORDSYS.ORD_STR_LIST default null),
```

### Description

Overlays a text watermark onto a source image and writes it to a destination image object.

### Parameters

**added_text**
The watermark text stored in a string to be added to the source image.

**dest**
The destination ORDImage object for the watermarked image.

**logging**
A string that contains information about any unexpected behavior that occurred during the watermarking operation. If the operation is successful, an empty string is returned. Otherwise, this method returns a string that describes the unexpected behavior. For example: if watermark text is so long that it is truncated, this string is returned: `WARNING: text is too long and truncated.`

**watermark_properties**
A string list of name-value pairs that define the attributes of the watermark text to be overlaid, including: font_name, font_style, font_size, text_color, position_x, position_y, and transparency. See Table D–2 for the complete list of watermark properties.

### Usage Notes

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).

See Section D.6 for more information about watermarking operations and watermark properties.

### Pragmas

None.

### Exceptions

ORDImageExceptions.DATA_NOT_LOCAL

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

ORDImageExceptions.NULL_DESTINATION

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

ORDImageExceptions.NULL_LOCAL_DATA

This exception is raised if you call the applyWatermark( ) method and the value of the destination image source.localData attribute is NULL. This exception is also raised if you call the applyWatermark( ) method and the value of the source image source.local attribute is 1 or NULL, and the value of the source.localData attribute is NULL.

ORDImageExceptions.NULL_SOURCE

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

See Appendix G for more information about these exceptions.

## Examples

Add watermark text to a source image:

```
-- add text as watermark to source image
DECLARE
  source_image ORDSYS.ORDImage;
  added_text varchar2(200);
  dest_image ORDSYS.ORDImage;
  prop ordsys.ord_str_list;
  logging VARCHAR2(2000);
BEGIN
  select product_photo into source_image from pm.online_media
    where product_id = 4001;
  select product_photo into dest_image from pm.online_media
    where product_id = 4003 for update;

  added_text := 'Oracle Multimedia © 2009';
  -- specify properties
  prop := ordsys.ord_str_list(
                  'font_name=Times New Roman',
                  'font_style=bold',
                  'font_size=50',
                  'text_color=red',
                  'position_x=100',
                  'position_y=100',
                  'transparency=0.6');

  -- add text watermark to source image
  source_image.applyWatermark(added_text, dest_image, logging, prop);
  update pm.online_media set product_photo = dest_image where product_id = 4003;
  commit;
EXCEPTION
  WHEN OTHERS THEN
      RAISE;
END;
/
```

where:

- product_id 4001: the source image (in JPEG format) stored in the table.
- product_id 4003: the resulting watermarked image stored in the table.

## checkProperties( )

### Format

checkProperties( ) RETURN BOOLEAN;

### Description

Verifies that the properties stored in attributes of the image object match the properties of the image. Do not use this method for foreign images (those formats not natively supported by Oracle Multimedia).

### Parameters

None.

### Usage Notes

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

### Pragmas

None.

### Exceptions

ORDImageExceptions.NULL_SOURCE

This exception is raised when the value of the ORDImage.source attribute is `NULL`.

See Appendix G for more information about this exception.

### 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 the source data is stored locally or not.

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

## Pragmas

None.

## Exceptions

ORDImageExceptions.NULL_SOURCE

This exception is raised when the value of the ORDImage.source attribute is `NULL`.

See Appendix G for more information about this exception.

## 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;
/
```

## getMetadata( )

### Format

getMetadata(metadataType IN VARCHAR2 DEFAULT 'ALL' ) RETURN XMLSequenceType ;

### Description

Extracts the specified types of metadata from the image and returns an array of schema-valid XML documents. If no matching metadata is found, an empty array is returned.

### Parameters

**metadataType**
A string that specifies the types of embedded metadata to extract. Valid values are: `ALL`, `ORDIMAGE`, `XMP`, `EXIF`, and `IPTC-IIM`. The default value is `ALL`.

### Usage Notes

When the value of the input parameter metadataType is `ALL`, and two or more types of supported metadata are present in the image, this method returns several XML documents, one for each type of metadata found. For other values of the input parameter, the method returns zero or one XML document.

Each document is stored as an instance of XMLType, and is based on one of the metadata schemas. Use the XQuery function `fn:namespace-uri` to determine the type of metadata represented in that document.

See Appendix F for a description of the supported metadata schemas.

**See Also:**

- *Oracle Multimedia User's Guide* for more information about the metadata feature
- *Oracle XML DB Developer's Guide* for more information about XQuery functions

### Pragmas

None.

### Exceptions

ORDImageExceptions.NULL_LOCAL_DATA

This exception is raised when source.localData is NULL.

ORDImageExceptions.NULL_SOURCE

This exception is raised when the value of the ORDImage.source attribute is `NULL`.

See Appendix G for more information about these exceptions.

### Examples

Extract the embedded metadata, and return an array of schema-valid XML documents:

```
DECLARE
  image ORDSYS.ORDImage;
  metav XMLSequenceType;
```

```
  ns    varchar2(4000);
BEGIN
  SELECT product_photo
  INTO image
  FROM pm.online_media
  WHERE product_id = 3106;

  metav := image.getMetadata('ALL');

  -- print the namespace of each metadata document
  FOR i in 1..metav.count LOOP
    select xmlcast(xmlquery('fn:namespace-uri($x)'
          passing metav(i) as "x" returning content) as varchar2(4000))
      into ns from dual;
    DBMS_OUTPUT.PUT_LINE('namespace: ' || ns);
  END LOOP;

  EXCEPTION
  WHEN ORDSYS.ORDImageExceptions.NULL_LOCAL_DATA THEN
    DBMS_OUTPUT.PUT_LINE('source local data is null');
  WHEN ORDSYS.ORDImageExceptions.NULL_SOURCE THEN
    DBMS_OUTPUT.PUT_LINE('source is null');
  WHEN OTHERS THEN
    RAISE;
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 source.localData attribute (of the embedded ORDSource object) within the database.

## Parameters

**ctx**
The source plug-in context information. This parameter must be allocated and initialized to NULL. If you are using a user-defined source plug-in, call the openSource( ) method. (See Section 5.2.)

## 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 before calling the import( ) method.

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.

If the value of the source.srcType attribute is `FILE`, the source.srcLocation attribute contains the name of a database directory object that contains the file to be imported, and the source.srcName attribute contains the name of the file to be imported. You must ensure that the directory for the external source location exists or is created before you use this method.

The import( ) method reads only from a database directory object that the user has privilege to access. That is, you can access a directory object that you have created using the SQL statement CREATE DIRECTORY, or one to which you have been granted READ access.

For example, the following SQL*Plus commands create a directory object and grant the user `ron` permission to read any file within the directory `c:\mydir\work`:

```
CONNECT sys as sysdba
Enter password: password
CREATE OR REPLACE DIRECTORY FILE_DIR AS 'c:\mydir\work';
GRANT READ ON DIRECTORY FILE_DIR TO ron;
```

See Section 5.1 for more information about directory and table definitions.

If the value of the source.srcType attribute is `HTTP`, the source.srcLocation attribute contains the base URL needed to find the directory that contains the object to be imported, and the source.srcName attribute contains the name of the object to be imported.

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 Linux and UNIX, setting the environment variable http_proxy to a URL specifies that the UTL_HTTP package must 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 not be used for URLs in the specified domain.

**See Also:**

*Oracle Database PL/SQL Packages and Types Reference* for more information about the UTL_HTTP package

If the value of the source.srcType attribute is a user-defined name, the source.srcLocation attribute contains an identifier string required to access the user-defined object to be imported, and the source.srcName attribute contains the name of the object to be imported.

## Pragmas

None.

## Exceptions

ORDImageExceptions.NULL_SOURCE

This exception is raised when the value of the ORDImage.source attribute is `NULL`.

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 G 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 parameter must be allocated and initialized to NULL. If you are using a user-defined source plug-in, call the openSource( ) method. (See Section 5.2.)

**source_type**
The type of the source image data. (See Table 2–1.)

**source_location**
The location from which the source image data is to be imported. (See Table 2–2.)

**source_name**
The name of the source image data. (See Table 2–3.)

## Usage Notes

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

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.

If the value of the source_type parameter is `FILE`, the source_location parameter contains the name of a database directory object that contains the file to be imported, and the source_name parameter contains the name of the file to be imported. You must ensure that the directory indicated by the source_location parameter exists or is created before you use this method.

The importFrom( ) method reads only from a database directory object that the user has privilege to access. That is, you can access a directory object that you have created using the SQL statement CREATE DIRECTORY, or one to which you have been granted READ access.

For example, the following SQL*Plus commands create a directory object and grant the user `ron` permission to read any file within the directory `c:\mydir\work`:

```
CONNECT sys as sysdba
Enter password: password
CREATE OR REPLACE DIRECTORY FILE_DIR AS 'c:\mydir\work';
GRANT READ ON DIRECTORY FILE_DIR TO ron;
```

See Section 5.1 for more information about directory and table definitions.

If the value of the source_type parameter is `HTTP`, the source_location parameter contains the base URL needed to find the directory that contains the object to be imported, and the source_name parameter contains the name of the object to be imported.

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 Linux and UNIX, setting the environment variable http_proxy to a URL specifies that the UTL_HTTP package must 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 not be used for URLs in the specified domain.

> **See Also:**
>
> *Oracle Database PL/SQL Packages and Types Reference* for more information about the UTL_HTTP package

If the value of the source_type parameter is a user-defined name, the source_location parameter contains an identifier string required to access the user-defined object to be imported, and the source_name parameter contains the name of the object to be imported.

## Pragmas

None.

## Exceptions

ORDImageExceptions.NULL_SOURCE

This exception is raised when the value of the ORDImage.source 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.

See Appendix G 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.

You can change one or more of the image attributes shown in Table 5–1.

***Table 5–1 Image Processing Operators***

| Operator Name | Usage | Values |
|---|---|---|
| compressionFormat | Forces output to the specified compression format if it is supported by the output file format. (See Section D.2.3.) | 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. (See Section D.2.4.) | MAXCOMPRATIO, MAXINTEGRITY, LOWCOMP, MEDCOMP, HIGHCOMP, or an integer between 0 and 100 |
| contentFormat | Determines the format of the image content. (See Section D.2.2.) | See Section D.2.2 for values |
| contrast | Adjusts image contrast. (See Section D.3.1.)[1] | nonnegative FLOAT[2],<br>nonnegative FLOAT FLOAT FLOAT[3],<br>nonnegative FLOAT FLOAT[2],<br>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. (See Section D.3.2.) | nonnegative INTEGER INTEGER INTEGER INTEGER maximum value is `2147483648` |
| fileFormat | Forces the output to specified file format. (See Section D.2.1.) | BMPF, CALS, GIFF, JFIF, PBMF, PGMF, PICT, PNGF, PNMF, PPMF, RASF, RPIX, TGAF, TIFF, WBMP |

***Table 5–1 (Cont.) Image Processing Operators***

| Operator Name | Usage | Values |
|---|---|---|
| fixedScale | Scales an image to a specified size in pixels (width, height); cannot be combined with other scale operators. (See Section D.3.10.1.) | positive INTEGER INTEGER |
| flip | Places the scanlines of an image in inverse order -- swapped top to bottom. (See Section D.3.3.) | No arguments |
| gamma | Adjusts gamma (brightness) of an image. (See Section D.3.4.)[1] | positive FLOAT[2]<br>positive FLOAT FLOAT FLOAT[5] |
| maxScale | Scales an image to a specified size in pixels (width, height), while maintaining the aspect ratio; cannot be combined with other scale operators. (See Section D.3.10.2.) | positive INTEGER INTEGER |
| mirror | Places columns of an image in reverse order -- swapped left to right. (See Section D.3.5.) | No arguments |
| nometadata | Processes an image without encoding the image's metadata in the resulting image. (See Section D.3.6.) | No arguments |
| page | Selects a page from a multipage file; for use with TIFF only; first page is 0 (zero). (See Section D.3.7.) | nonnegative INTEGER |
| quantize | Specifies how image quantization is to be performed when reducing image bit depth. (See Section D.3.8.) | ERRORDIFFUSION (default), ORDEREDDITHER, THRESHOLD, MEDIANCUT |
| rotate | Rotates an image within the image plane by the angle specified. (See Section D.3.9.)[1] | FLOAT |
| scale | Uniformly scales an image by a given factor (for example, 0 .5 or 2.0); cannot be combined with other scale operators. (See Section D.3.10.3.)[1] | positive FLOAT |
| sharpen | Sharpens an image by a given kernel type and gain factor (for example, 0 and 2.0); can be combined with scale operators. (See Section D.3.11.)[1] | positive INTEGER<br>positive FLOAT |

*Table 5–1 (Cont.) Image Processing Operators*

| Operator Name | Usage | Values |
|---|---|---|
| tiled | Forces output image to be tiled; for use with TIFF only. (See Section D.3.12.) | No arguments |
| transparencyFillColor | Fills the transparent regions of PNG image file formats with color. (See Section D.3.13.) | BLACK, BLUE, CYAN, DARK_GRAY, GRAY, GREEN, LIGHT_GRAY MAGENTA, ORANGE, PINK, RED, WHITE, YELLOW, or a nonnegative INTEGER INTEGER INTEGER |
| xScale | Scales an image on the X-axis by a given factor (default is 1); image is non-uniformly scaled; can be combined with the yScale operator only; cannot be combined with any other scale operators. (See Section D.3.10.4.)[1] | positive FLOAT |
| yScale | Scales the image on theY-axis scale by a given factor (default is 1); non-uniformly scales image; can be combined with the xScale operator only; cannot be combined with any other scale operators. (See Section D.3.10.5.)[1] | positive FLOAT |

[1] Enclose floating-point arguments with double quotation marks to ensure correct Globalization Support interpretation.
[2] Specifies the percent contrast enhancement to be applied to all bands (GRAY or RGB)
[3] Specifies the percent contrast enhancement to be applied to each band (RGB only)
[4] Specifies the bounds for contrast enhancement to be applied to each band (RGB only)
[5] Specifies separate gamma values to be applied to each band (RGB only)

**Note:** To ensure that floating-point values are interpreted according to the NLS_TERRITORY setting for the session, surround the value with double quotation marks (""). For example, use `'scale="0.7"'` in the AMERICAN territory, and `'scale="0,7"'` in the FRENCH territory.

Table 5–2 shows additional changes that can be made only to raw pixel and foreign images.

*Table 5–2 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. (See Section D.4.1.) | 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). This operator affects the source image, not the destination; RPIX only. (See Section D.4.4.) | positive INTEGER,[1]<br>positive INTEGER INTEGER INTEGER[2] |
| pixelOrder | Forces pixel direction. If NORMAL, then the leftmost pixel appears first in the image. RPIX only. (See Section D.4.2.) | NORMAL (default), REVERSE |
| scanlineOrder | Forces scanline direction. If NORMAL, then the top scanline appears first in the image. RPIX and BMPF only. (See Section D.4.3.) | NORMAL (default), INVERSE |

[1] Specifies that a single band is to be selected from the input image and that band is to be used to create a grayscale output image

[2] Specifies that three bands are to be selected from the input image and those bands are to specify the red, green, and blue bands of an RGB output image

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

**See Also:**

*Oracle Multimedia User's Guide* for more information about Globalization Support

## 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).

ORDImageExceptions.NULL_SOURCE

This exception is raised when the value of the ORDImage.source attribute is `NULL`.

See Appendix G for more information about these exceptions.

## 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:** 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

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 process( ), specifically Table 5–1 and Table 5–2, for information about image processing operators.

See Appendix D for more information about processCopy( ) method operators.

### Pragmas

None.

### Exceptions

ORDImageExceptions.DATA_NOT_LOCAL

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

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

ORDImageExceptions.NULL_SOURCE

This exception is raised if the value of the source attribute for the source image is NULL.

See Appendix G for more information about these exceptions.

## 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;
/
```

## putMetadata( )

### Format

```
putMetadata(xmlData          IN NOCOPY XMLType,
            metadataType IN VARCHAR2 DEFAULT 'XMP'
            encoding         IN VARCHAR2 DEFAULT 'UTF-8');
```

### Description

Accepts a schema-valid XML document and creates a binary packet suitable for embedding in the target image file format. The packet is encoded according to the value of the encoding parameter. If the value of the metadataType parameter is `XMP`, this method writes a new XMP packet to the image, replacing any existing XMP packets.

### Parameters

**xmlData**
The XMLType that contains a schema-valid XML document for the indicated metadataType. If the value of the metadataType parameter is `XMP`, the root element should contain a well-formed RDF document.

**metadataType**
A string that specifies the type of metadata to write. The valid value is `XMP`; it is also the default.

**encoding**
The character encoding to be used in the image file. Valid values are: `UTF-8`, `UTF-16`, `UTF-16BE`, and `UTF-16LE`. The default is `UTF-8`.

### Usage Notes

The binary metadata packet generated from the same xmlData input may have different sizes for different encodings. Different image file formats support different encodings, and may restrict the binary metadata packet size. The restrictions of the supported image formats are as follows:

- GIF89a supports UTF-8 encoding only.
- JPEG requires a binary packet size of less than 65502 bytes.
- TIFF requires a binary packet size of less than 4 gigabytes.

> **See Also:**
>
> *Oracle Multimedia User's Guide* for more information about the metadata feature

### Pragmas

None.

### Exceptions

ORDImageExceptions.DATA_NOT_LOCAL

This exception is raised when the data is not local (the source.local attribute is 0.)

ORDImageExceptions.NULL_LOCAL_DATA

This exception is raised when source.localData is NULL.

ORDImageExceptions.NULL_SOURCE

This exception is raised when the value of the ORDImage.source attribute is NULL.

See Appendix G for more information about these exceptions.

## Examples

Replace an ORDImage object in the table pm.online_media with a new ORDImage object, with updated XMP metadata:

```
DECLARE
  image     ORDSYS.ORDImage;
  xmlData   XMLType;
BEGIN
  SELECT product_photo
  INTO image
  FROM pm.online_media
  WHERE product_id = 3106 FOR UPDATE;


  xmlData := xmltype(
       '<xmpMetadata xmlns="http://xmlns.oracle.com/ord/meta/xmp">' ||
       '<rdf:RDF xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"' ||
       '  xmlns:dc="http://purl.org/dc/elements/1.1/">' ||
       '<dc:rights>' ||
       '  <rdf:Alt>' ||
       '    <rdf:li xml:lang="en-us">' ||
       '      Oracle Corporation' ||
       '    </rdf:li>' ||
       '  </rdf:Alt>'||
       '</dc:rights>' ||
       '</rdf:RDF>' ||
       '</xmpMetadata>', 'http://xmlns.oracle.com/ord/meta/xmp');


  image.putMetadata(xmlData, 'xmp', 'utf-8');


  UPDATE pm.online_media
  SET product_photo = image
  WHERE product_id=3106;
  COMMIT;

  EXCEPTION
    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('source.localData attribute is null');
    WHEN ORDSYS.ORDImageExceptions.NULL_SOURCE THEN
      DBMS_OUTPUT.PUT_LINE('source is null');
    WHEN OTHERS THEN
      RAISE;
    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 so as to determine the image characteristics.

Do not call this method 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, unless this method is called implicitly.

This method sets this 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 on)
- Image type (monochrome and so on)
- Compression type (JPEG, LZW, and so on)
- 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 value of the source.local attribute is `1` or `NULL` and the value of the source.localData attribute is `NULL`.

ORDImageExceptions.NULL_SOURCE

This exception is raised when the value of the ORDImage.source attribute is `NULL`.

See Appendix G for more information about these exceptions.

## 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 Oracle Multimedia 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 Oracle Multimedia detects an unknown file format, it does not implicitly set the properties.

See Appendix E for information about 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 about how to interpret the bits in the file or, Oracle Multimedia 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 inTable 5–3.

***Table 5–3 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 Oracle Multimedia 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. The default is zero. |
| DefaultChannelSelection | INTEGER<br>or<br>INTEGER,<br>INTEGER,<br>INTEGER | Specifies which bands in a multiband image are interpreted as color channels when the image is read or processed. If a single integer is specified, the image is treated as a grayscale image consisting of the data in the specified band only. If three integers are specified, the image is 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.<br>For example:<br>■ Specify "DefaultChannelSelection = 1" to cause the first band of the image to be interpreted as a grayscale image.<br>■ Specify "DefaultChannelSelection = 4" to cause the fourth band of the image to be interpreted as a grayscale image.<br>■ 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.<br>■ 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. The 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:<br>■ BIP (default) Band Interleaved by Pixel<br>■ BIL Band Interleaved by Line<br>■ 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. The 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. |
| 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:". The default is blank and fileFormat is set to "OTHER". |
| Width | INTEGER | Specifies the width of the image in pixels. The 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 value of the description parameter is NULL.

See Appendix G for more information about this exception.

## 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;
/
```

# 6

# Oracle Multimedia ORDVideo Object Type

Oracle Multimedia provides the ORDVideo object type, which supports the storage and management of video data.

The ORDVideo object type is defined in the `ordvspec.sql` file. After installation, this file is available in the Oracle home directory at:

*<ORACLE_HOME>*`/ord/im/admin` (on Linux and UNIX)

*<ORACLE_HOME>*`\ord\im\admin` (on Windows)

This chapter contains these sections:



## 6.1 ORDVideo Object Examples

The examples in this chapter use the ONLINE_MEDIA table in the Product Media (PM) sample schema. To replicate the examples on your computer, begin with the examples shown in the reference pages for the ORDVideo constructors and the import( ) and importFrom( ) methods. Then, substitute your video files for those in the examples.

The example for the constructor ORDVideo for BLOBs uses the test video table TVID (see TVID Table Definition).

---

**Note:** If you manipulate the video data itself (by either directly modifying the BLOB or changing the external source), 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.

---

**See Also:**

*Oracle Database Sample Schemas* for more information about the PM schema

### 6.1.1 Directory Definition for ORDVideo Object Examples

Issue the following statements before executing the examples, where `c:\mydir\work` is the directory where the user `ron` can find the video data:

```
CONNECT sys as sysdba
CREATE OR REPLACE DIRECTORY FILE_DIR as 'c:\mydir\work';
GRANT READ ON DIRECTORY FILE_DIR TO 'ron';
```

## 6.2 Important Notes for ORDVideo Methods

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 must 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 must 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 must 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 use the ctx argument, but if you code as previously described, your application should work with current or future source or format plug-ins.

---

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. The attributes for this object type are defined as follows in the `ordvspec.sql` file:

```
--------------------
-- TYPE 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,
```

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. Oracle recommends that you not write to this attribute directly.

## ORDVideo Constructors

This section describes these Oracle Multimedia constructor functions:

## ORDVideo for BLOBs

### Format

```
ORDVideo(SELF IN OUT NOCOPY ORDSYS.ORDVideo,
         data IN BLOB,
         setproperties IN INTEGER DEFAULT 0)
RETURN SELF AS RESULT
```

### Description

Constructs an ORDVideo object from a BLOB. The data stored in the BLOB is copied into the ORDVideo object when the constructed ORDVideo object is inserted or updated into a table.

### Parameters

**data**
Video content stored in a BLOB.

**setproperties**
Indicator flag that determines whether the setProperties( ) method is called in the constructor. If the value is `1`, the setProperties( ) method is called. If the value is `0`, the method is not called. The default is `0`.

### Pragmas

None.

### Exceptions

None.

### Usage Notes

Use this constructor to create an ORDVideo object when the audio content is stored in either a temporary or a persistent BLOB.

### Examples

Create an ORDVideo object from a BLOB object and insert it into the table.

> **Note:** The user who runs this statement must have the SELECT privilege on the table `tvid` created by the user `mediauser`.
>
> See Section 7.2.5 for a definition of the test video table TVID.

```
INSERT INTO pm.online_media (product_id, product_video)
  SELECT 2004, ORDSYS.ORDVideo(t.vid) FROM mediauser.tvid t WHERE t.n = 1;
```

# ORDVideo for Other Sources

## Format

```
ORDVideo( SELF IN OUT NOCOPY ORDSYS.ORDVideo,
                  source_type      IN VARCHAR2 DEFAULT 'LOCAL',
                  source_location  IN VARCHAR2 DEFAULT NULL,
                  source_name      IN VARCHAR2 DEFAULT NULL,
                  setproperties  IN INTEGER DEFAULT 0
) RETURN SELF AS RESULT
```

## Description

Constructs an ORDVideo object from a specific source. By default, an empty object with a local source is constructed.

## Parameters

**source_type**
The type of the source video data. Valid values are: `FILE`, `HTTP`, `LOCAL`, or user-defined. The default is `LOCAL`.

The parameter value `LOCAL` indicates that the data is stored in Oracle Database. The parameter value `LOCAL` is never stored in the srcType attribute. Rather, this parameter value indicates that the data is stored in Oracle Database in the localData attribute. (See Section 2.2 for a description of the ORDSource object.)

**source_location**
The location from which the source video data is to be imported. (See Table 2–2.)

**source_name**
The name of the source video data. (See Table 2–3.)

**setproperties**
Indicator flag that determines whether the setProperties( ) method is called in the constructor. If the value is `1`, the setProperties( ) method is called. If the value is `0`, the method is not called. The default is `0`.

## Pragmas

None.

## Exceptions

None.

## Usage Notes

Use this constructor to create an ORDVideo object when the video content is not stored in a BLOB, as in any of these situations:

The content is in a local file.
The content is referenced by a URL.
The content is not available when the ORDVideo object is created.

## Examples

Create an ORDVideo object from a specified source:

```
-- Construct an ORDVideo object from an external file and insert it into the
table.
INSERT INTO pm.online_media (product_id, product_video)
  VALUES (2030, ORDSYS.ORDVideo('FILE', 'FILE_DIR', 'speakers.rm'));

-- Initialize an ORDVideo object in the table.
INSERT INTO pm.online_media (product_id, product_video)
  VALUES(2004, ORDSYS.ORDVideo());
```

# ORDVideo Methods

This section presents reference information about the Oracle Multimedia methods used specifically for video data manipulation.

Chapter 2 presents reference information about the Oracle Multimedia 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.

This section describes these methods:

- setVideoDuration( ) on page 6-52

**See Also:**

*Oracle Database Concepts* for more information about object types and methods

# checkProperties( )

## Format

checkProperties(ctx IN OUT RAW) RETURN BOOLEAN;

## Description

Checks all the properties of the stored video data, including these 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. (See Section 6.2.)

## 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.NULL_SOURCE

This exception is raised when the value of the ORDVideo.source attribute is `NULL`.

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.

See Appendix G for more information about these exceptions.

## 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 this 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. (See Section 6.2.)

**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 Also:**

*Oracle Multimedia User's Guide* for more information about extending support for media formats

## 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.

ORDVideoExceptions.NULL_SOURCE

This exception is raised when the value of the ORDVideo.source attribute is `NULL`.

See Appendix G for more information about these exceptions.

## 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. (See Section 6.2.)

**name**
The name of the attribute.

## Usage Notes

None.

## Pragmas

None.

## Exceptions

ORDVideoExceptions.NULL_SOURCE

This exception is raised when the value of the ORDVideo.source attribute is `NULL`.

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.

See Appendix G for more information about these exceptions.

## 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 is 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. (See Section 6.2.)

**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.

ORDVideoExceptions.NULL_SOURCE

This exception is raised when the value of the ORDVideo.source attribute is `NULL`.

See Appendix G 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. (See Section 6.2.)

## Usage Notes

None.

## Pragmas

None.

## Exceptions

ORDSourceExceptions.INCOMPLETE_SOURCE_INFORMATION

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

ORDVideoExceptions.NULL_SOURCE

This exception is raised when the value of the ORDVideo.source attribute is `NULL`.

See Appendix G for more information about these exceptions.

## Examples

See the example in import( ) on page 6-30.

## 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.

See Appendix G for more information about this exception.

### Examples

See the example in setDescription( ) on page 6-39.

# 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 of the format attribute is `NULL`.

See Appendix G for more information about this exception.

## 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 the source.localData attribute (of the embedded ORDSource object) within the database.

## Parameters

**ctx**
The source plug-in context information. This parameter must be allocated and initialized to NULL. If you are using a user-defined source plug-in, call the openSource( ) method. (See Section 6.2.)

## 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 before 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.

If the value of the source.srcType attribute is `FILE`, the source.srcLocation attribute contains the name of a database directory object that contains the file to be imported, and the source.srcName attribute contains the name of the file to be imported. You must ensure that the directory for the external source location exists or is created before you use this method.

The import( ) method reads only from a database directory object that the user has privilege to access. That is, you can access a directory object that you have created using the SQL statement CREATE DIRECTORY, or one to which you have been granted READ access.

For example, the following SQL*Plus commands create a directory object and grant the user `ron` permission to read any file within the directory `c:\mydir\work`:

```
CONNECT sys as sysdba
Enter password: password
CREATE OR REPLACE DIRECTORY FILE_DIR AS 'c:\mydir\work';
GRANT READ ON DIRECTORY FILE_DIR TO ron;
```

See Section 6.1 for more information about directory and table definitions.

If the value of the source.srcType attribute is `HTTP`, the source.srcLocation attribute contains the base URL needed to find the directory that contains the object to be imported, and the source.srcName attribute contains the name of the object to be imported.

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 Linux and UNIX, setting the environment variable http_proxy to a URL specifies that the UTL_HTTP package must 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 not be used for URLs in the specified domain.

**See Also:**

*Oracle Database PL/SQL Packages and Types Reference* for more information about the UTL_HTTP package

If the value of the source.srcType attribute is a user-defined name, the source.srcLocation attribute contains an identifier string required to access the user-defined object to be imported, and the source.srcName attribute contains the name of the object to be imported.

## 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.

ORDSourceExceptions.NULL_SOURCE

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

ORDVideoExceptions.NULL_SOURCE

This exception is raised when the value of the ORDVideo.source attribute is NULL.

See Appendix G 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 the source.localData attribute (of the embedded ORDSource object) within the database.

## Parameters

**ctx**
The source plug-in context information. This parameter must be allocated and initialized to `NULL`. If you are using a user-defined source plug-in, call the openSource( ) method. (See Section 6.2.)

**source_type**
The type of the source video data. (See Table 2–1.)

**source_location**
The location from which the source video data is to be imported. (See Table 2–2.)

**source_name**
The name of the source video data. (See Table 2–3.)

## Usage Notes

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

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.

If the value of the source_type parameter is `FILE`, the source_location parameter contains the name of a database directory object that contains the file to be imported, and the source_name parameter contains the name of the file to be imported. You must ensure that the directory indicated by the source_location parameter exists or is created before you use this method.

The importFrom( ) method reads only from a database directory object that the user has privilege to access. That is, you can access a directory object that you have created using the SQL statement CREATE DIRECTORY, or one to which you have been granted READ access.

For example, the following SQL*Plus commands create a directory object and grant the user `ron` permission to read any file within the directory `c:\mydir\work`:

```
CONNECT sys as sysdba
Enter password: password
CREATE OR REPLACE DIRECTORY FILE_DIR AS 'c:\mydir\work';
```

```
GRANT READ ON DIRECTORY FILE_DIR TO ron;
```

See Section 6.1 for more information about directory and table definitions.

If the value of the source_type parameter is HTTP, the source_location parameter contains the base URL needed to find the directory that contains the object to be imported, and the source_name parameter contains the name of the object to be imported.

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 Linux and UNIX, setting the environment variable http_proxy to a URL specifies that the UTL_HTTP package must 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 not be used for URLs in the specified domain.

> **See Also:**
>
> *Oracle Database PL/SQL Packages and Types Reference* for more information about the UTL_HTTP package

If the value of the source_type parameter is a user-defined name, the source_location parameter contains an identifier string required to access the user-defined object to be imported, and the source_name parameter contains the name of the object to be imported.

## 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.NULL_SOURCE exception

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

ORDVideoExceptions.NULL_SOURCE

This exception is raised when the value of the ORDVideo.source attribute is NULL.

See Appendix G 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.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. (See Section 6.2.)

**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 Also:**
>
> *Oracle Multimedia User's Guide* for more information about extending support for media formats

### Pragmas

None.

## Exceptions

ORDVideoExceptions.METHOD_NOT_SUPPORTED

This exception is raised when the video plug-in does not support the method or the plug-in is not found.

ORDVideoExceptions.NULL_SOURCE

This exception is raised when the value of the ORDVideo.source attribute is `NULL`.

ORDVideoExceptions.VIDEO_PLUGIN_EXCEPTION

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

See Appendix G for more information about these exceptions.

## 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 of the knownBitRate parameter is `NULL`.

ORDVideoExceptions.NULL_SOURCE

This exception is raised when the value of the ORDVideo.source attribute is `NULL`.

See Appendix G for more information about these exceptions.

### Examples

See the example in setFrameSize( ) on page 6-44.

# 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 of the knownCompressionType parameter is `NULL`.

ORDVideoExceptions.NULL_SOURCE

This exception is raised when the value of the ORDVideo.source attribute is `NULL`.

See Appendix G for more information about these exceptions.

## Examples

See the example in setFrameSize( ) on page 6-44.

## 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 Multimedia make use of this description attribute to present video data to users.

Calling this method implicitly calls the setUpdateTime( ) method.

### Pragmas

None.

### Exceptions

ORDVideoExceptions.NULL_SOURCE

This exception is raised when the value of the ORDVideo.source attribute is NULL.

See Appendix G for more information about this exception.

### 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 of the knownFormat parameter is NULL.

ORDVideoExceptions.NULL_SOURCE

This exception is raised when the value of the ORDVideo.source attribute is NULL.

See Appendix G for more information about these exceptions.

## 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 of the knownFrameRate parameter is `NULL`.

ORDVideoExceptions.NULL_SOURCE

This exception is raised when the value of the ORDVideo.source attribute is `NULL`.

See Appendix G for more information about these exceptions.

## Examples

See the example in setFrameSize( ) on page 6-44.

## 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 of the knownFrameResolution parameter is `NULL`.

ORDVideoExceptions.NULL_SOURCE

This exception is raised when the value of the ORDVideo.source attribute is `NULL`.

See Appendix G for more information about these exceptions.

### Examples

See the example in setFrameSize( ) on page 6-44.

# 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 of either the knownWidth or the knownHeight parameter is `NULL`.

ORDVideoExceptions.NULL_SOURCE

This exception is raised when the value of the ORDVideo.source attribute is `NULL`.

See Appendix G for more information about these exceptions.

## 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

ORDVideoExceptions.NULL_SOURCE

This exception is raised when the value of the ORDVideo.source attribute is NULL.

See Appendix G for more information about this exception.

## 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('bit 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 of the knownNumberOfColors parameter is `NULL`.

ORDVideoExceptions.NULL_SOURCE

This exception is raised when the value of the ORDVideo.source attribute is `NULL`.

See Appendix G for more information about these exceptions.

## Examples

See the example in setFrameSize( ) on page 6-44.

## 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 of the knownNumberOfFrames parameter is `NULL`.

ORDVideoExceptions.NULL_SOURCE

This exception is raised when the value of the ORDVideo.source attribute is `NULL`.

See Appendix G for more information about these exceptions.

### Examples

See the example in setFrameSize( ) on page 6-44.

# 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 these 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. (See Section 6.2.)

**setComments**
A Boolean value that indicates whether 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.NULL_SOURCE

This exception is raised when the value of the ORDVideo.source attribute is `NULL`.

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.

See Appendix G for more information about these exceptions.

## 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 of the knownVideoDuration parameter is `NULL`.

ORDVideoExceptions.NULL_SOURCE

This exception is raised when the value of the ORDVideo.source attribute is `NULL`.

See Appendix G for more information about these exceptions.

## Examples

See the example in setFrameSize( ) on page 6-44.

# 7

# Oracle Multimedia Relational Interface

Oracle Multimedia provides a relational interface that consists of a set of static methods for each of these Oracle Multimedia objects:

- ORDAudio
- ORDDoc
- ORDImage
- ORDVideo

Because these are static methods, no objects are instantiated. So, data is passed by method arguments rather than by object attributes.

The static methods for these four Oracle Multimedia objects are defined in the `ordaspec.sql`, `orddspec.sql`, `ordispec.sql`, and `ordvspec.sql` files, respectively. After installation, these files are available in the Oracle home directory at:

*<ORACLE_HOME>*`/ord/im/admin` (on Linux and UNIX)

*<ORACLE_HOME>*`\ord\im\admin` (on Windows)

This chapter includes these sections:



## 7.1 Purpose of the Oracle Multimedia Relational Interface

Application developers, who created multimedia applications without using the Oracle Multimedia object types to store and manage media data in relational tables, and who do not want to migrate their existing multimedia applications to use Oracle Multimedia objects, can use the Oracle Multimedia relational interface for managing

their media data. The Oracle Multimedia relational interface consists of a set of methods for these operations:

- Extracting information directly from media data, either as 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 Oracle Multimedia relational interface enables application developers to take advantage of Oracle Multimedia functions with only minimal changes to their applications, and all without having to change their schemas to the Oracle Multimedia objects to store their data.

## 7.2 Examples for Relational Methods

The examples of static methods for the relational interface (including methods common to all object types and methods that are unique to a particular object type) use this list of tables:

| Name | Purpose | Definition |
|---|---|---|
| TAUD | ORDAudio relational methods (and the common relational method export( )) | TAUD Table Definition |
| TDOC | ORDDoc relational methods (and the common relational method importFrom( )) | TDOC Table Definition |
| TIMG | ORDImage relational methods (and the common relational method importFrom( ) (all attributes)) | TIMG Table Definition |
| TVID | ORDVideo relational methods | TVID Table Definition |

When reading through the examples in this chapter, use the directory and table definitions that are provided in this section and with the example for each relational method.

---

**Note:** The tables described in this chapter are also used in the examples for these object constructors:

- ORDAudio for BLOBs ( on page 3-5)
- ORDDoc for BLOBs ( on page 4-5)
- ORDImage for BLOBs ( on page 5-5)
- ORDVideo for BLOBs ( on page 6-5)

---

### 7.2.1 Directory Definitions for Relational Examples

Some examples in this chapter use `mediauser` to represent the user, and `c:\mydir\work` to represent the directory specification where your test files can be located. See the example for each method for specific directory definitions for media data files and other details specific to that method.

### 7.2.2 TAUD Table Definition

The examples in Static Methods Unique to the ORDAudio Object Type Relational Interface use the audio table TAUD.

```
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)
LOB(aud) STORE AS SECUREFILE;

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;
```

### 7.2.3 TDOC Table Definition

The examples in Static Methods Unique to the ORDDoc Object Type Relational Interface use the document table TDOC.

```
CREATE TABLE tdoc(n             NUMBER,
                  document      BLOB,
                  attributes    CLOB,
                  mimetype      VARCHAR2(80),
                  format        VARCHAR2(80),
                  contentlength INTEGER)
LOB(document) STORE AS SECUREFILE;

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;
```

### 7.2.4 TIMG Table Definition

The examples in Static Methods Unique to the ORDImage Object Type Relational Interface use the image table TIMG.

```
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)
LOB(img) STORE AS SECUREFILE;

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;
```

### 7.2.5 TVID Table Definition

The examples in Static Methods Unique to the ORDVideo Object Type Relational Interface use the video table TVID.

```
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)
LOB(vid) STORE AS SECUREFILE;

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;
```

## 7.3 Important Notes for Relational Methods

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 must 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 must allocate the ctx structure and initialize it to NULL.

## Static Methods Common to All Object Types

This section presents reference information about these Oracle Multimedia common static methods, which are used for the relational interface:



The common static methods for the ORDAudio, ORDDoc, ORDImage, and ORDVideo relational interfaces are defined in the `ordaspec.sql`, `orddspec.sql`, `ordispec.sql`, and `ordvspec.sql` files, respectively.

> **Note:** The examples in this section assume that these tables exist: TAUD, TDOC, and TIMG.

# 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 provides native support only when the value of the source_type parameter is `FILE`. In this case, this method writes the data to a file within a directory that is accessible to Oracle Database. User-defined sources may support the export( ) method to provide WRITE access to other types of data stores.

## Parameters

**ctx**
The source plug-in context information. (See Section 7.3.)

**local_data**
The BLOB location that is being exported.

**source_type**
The type of the external source data. This parameter is not case sensitive. (See Table 2–1.)

**source_location**
The location to which the source data is to be exported. (See Table 2–2.)

**source_name**
The name of the object to which the source data is to be exported. (See Table 2–3.)

## 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` does not modify data stored in the BLOB.

When the source_type parameter has a value of `FILE`, the source_location parameter specifies the name of an Oracle directory object, and the source_name parameter specifies the name of the file to contain the data.

The export( ) method writes only to a database directory object that the user has privilege to access. That is, you can access a directory object that you have created using the SQL statement CREATE DIRECTORY, or one to which you have been granted READ and WRITE access.

For example, the following SQL*Plus commands create a directory object and grant the user `mediauser` permission to read and write to any file within the directory `c:\mydir\work`:

```
CONNECT sys as sysdba
Enter password: password
CREATE OR REPLACE DIRECTORY FILE_DIR AS 'c:\mydir\work';
GRANT READ,WRITE ON DIRECTORY FILE_DIR TO mediauser;
```

See Section 7.2 for more information about directory and table definitions.

## 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.IO_ERROR

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

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.

See Appendix G for more information about these exceptions.

## Examples

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

> **Note:** Replace *c:\mydir\work* with the directory specification where your exported file should be located, and replace *password* with the `sys` password.

```
CONNECT sys as sysdba
Enter password: password;
CREATE OR REPLACE DIRECTORY AUDIODIR AS 'c:\mydir\work';
GRANT READ, WRITE ON DIRECTORY AUDIODIR TO mediauser;

CONNECT mediauser
Enter password: password;

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 BLOB specified by the local_data parameter.

## Parameters

**ctx**
The source plug-in context information. This parameter must be allocated and initialized to NULL. If you are using a user-defined source plug-in, call the openSource( ) method. (See Section 7.3.)

**local_data**
The BLOB location to receive the data.

**source_type**
The type of the source data. (See Table 2–1.)

**source_location**
The location from which the source data is to be imported. (See Table 2–2.)

**source_name**
The name of the source data. (See Table 2–3.)

## Usage Notes

If the value of the source_type parameter is `FILE`, the source_location parameter contains the name of a database directory object that contains the file to be imported, and the source_name parameter contains the name of the file to be imported. You must ensure that the directory for the external source location exists or is created before you use this method.

The importFrom( ) method reads only from a database directory object that the user has privilege to access. That is, you can access a directory object that you have created using the SQL statement CREATE DIRECTORY, or one to which you have been granted READ access.

For example, the following SQL*Plus commands create a directory object and grant the user `mediauser` permission to read any file within the directory `c:\mydir\work`:

```
CONNECT sys as sysdba
Enter password: password
CREATE OR REPLACE DIRECTORY FILE_DIR AS 'c:\mydir\work';
GRANT READ ON DIRECTORY FILE_DIR TO mediauser;
```

See Section 7.2 for more information about directory and table definitions.

If the value of the source_type parameter is HTTP, the source_location parameter contains the base URL needed to find the directory that contains the object to be imported, and the source_name parameter contains the name of the object to be imported.

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 Linux and UNIX, setting the environment variable http_proxy to a URL specifies that the UTL_HTTP package must 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 not be used for URLs in the specified domain.

**See Also:**

*Oracle Database PL/SQL Packages and Types Reference* for more information about the UTL_HTTP package

If the value of the source_type parameter is a user-defined name, the source_location parameter contains an identifier string required to access the user-defined object to be imported, and the source_name parameter contains the name of the object to be imported.

## 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.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.

See Appendix G for more information about these exceptions.

## Examples

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

**Note:** Replace *c:\mydir\work* with the directory specification where your test files are located, replace *testimg.dat*, *testaud.dat*, and *testvid.dat* with the file specifications of your test files, and replace *password* with the system password.

```
CONNECT sys as sysdba
Enter password: password;
CREATE OR REPLACE DIRECTORY DOCDIR AS 'c:\mydir\work';
GRANT READ ON DIRECTORY DOCDIR TO mediauser;

CONNECT mediauser
Enter password: password;

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 BLOB specified by the local_data parameter.

## Parameters

**ctx**
The source plug-in context information. (See Section 7.3.)

**local_data**
The BLOB location to receive the data.

**source_type**
The type of the source data. (See Table 2–1.)

**source_location**
The location from which the source data is to be imported. (See Table 2–2.)

**source_name**
The name of the source data. (See Table 2–3.)

**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

If the value of the source_type parameter is `FILE`, the source_location parameter contains the name of a database directory object that contains the file to be imported, and the source_name parameter contains the name of the file to be imported. You must ensure that the directory for the external source location exists or is created before you use this method.

The importFrom( ) method reads only from a database directory object that the user has privilege to access. That is, you can access a directory object that you have created using the SQL statement CREATE DIRECTORY, or one to which you have been granted READ access.

For example, the following SQL*Plus commands create a directory object and grant the user `mediauser` permission to read any file within the directory `c:\mydir\work`:

```
CONNECT sys as sysdba
Enter password: password
CREATE OR REPLACE DIRECTORY FILE_DIR AS 'c:\mydir\work';
GRANT READ ON DIRECTORY FILE_DIR TO mediauser;
```

See Section 7.2 for more information about directory and table definitions.

If the value of the source_type parameter is `HTTP`, the source_location parameter contains the base URL needed to find the directory that contains the object to be imported, and the source_name parameter contains the name of the object to be imported.

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 Linux and UNIX, setting the environment variable http_proxy to a URL specifies that the UTL_HTTP package must 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 not be used for URLs in the specified domain.

**See Also:**

*Oracle Database PL/SQL Packages and Types Reference* for more information about the UTL_HTTP package

If the value of the source_type parameter is a user-defined name, the source_location parameter contains an identifier string required to access the user-defined object to be imported, and the source_name parameter contains the name of the object to be imported.

## 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.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.

See Appendix G for more information about these exceptions.

## Examples

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

**Note:** Replace *c:\mydir\work* with the directory specification where your test file is located, replace *testimg.dat* with the file specification of your test file, and replace *password* with the system password.

```
CONNECT sys as sysdba
Enter password: password;
CREATE OR REPLACE DIRECTORY IMAGEDIR AS 'c:\mydir\work';
GRANT READ ON DIRECTORY IMAGEDIR TO mediauser;

CONNECT mediauser
Enter password: password;

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 about these Oracle Multimedia static methods, which are unique to the ORDAudio relational interface:



The relational interface adds Oracle Multimedia support to audio data stored in BLOBs and BFILEs rather than in the ORDAudio object type. The static methods that are unique to the ORDAudio relational interface are defined in the `ordaspec.sql` file.

## 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. (See Section 7.3.)

**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`.

See Appendix G for more information about these exceptions.

### 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 these 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. (See Section 7.3.)

**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`.

See Appendix G for more information about these exceptions.

## 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;
/
```

# 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. (See Section 7.3.)

**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`.

See Appendix G for more information about these exceptions.

## 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 these 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. (See Section 7.3.)

**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`.

See Appendix G for more information about these exceptions.

## 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;
/
```

## Static Methods Unique to the ORDDoc Object Type Relational Interface

This section presents reference information about these Oracle Multimedia static methods, which are unique to the ORDDoc relational interface:



The relational interface adds Oracle Multimedia support to audio, image, video, and other heterogeneous media data stored in BLOBs and BFILEs rather than in the ORDDoc object type. The static methods that are unique to the ORDDoc relational interface are defined in the `orddspec.sql` file.

# 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. (See Section 7.3.)

**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`.

See Appendix G for more information about these exceptions.

## 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 these 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. (See Section 7.3.)

**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 of the content, 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

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`.

See Appendix G for more information about these exceptions.

## 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;
/
```

# 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. (See Section 7.3.)

**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`.

See Appendix G for more information about these exceptions.

## 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 these 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. (See Section 7.3.)

**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 of the content, 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

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`.

See Appendix G for more information about these exceptions.

## 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;
/
```

# Static Methods Unique to the ORDImage Object Type Relational Interface

This section presents reference information about these Oracle Multimedia static methods, which are unique to the ORDImage relational interface:



The relational interface adds Oracle Multimedia support to image data stored in BLOBs and BFILEs rather than in the ORDImage object type. The static methods that are unique to the ORDImage relational interface are defined in the `ordispec.sql` file.

## applyWatermark( ) image for BFILEs

### Format

```
applyWatermark(imageBfile    IN OUT NOCOPY BFILE,
               added_image     IN OUT NOCOPY BFILE,
               dest         IN OUT NOCOPY BLOB,
               logging OUT VARCHAR2,
               watermark_properties IN ordsys.ord_str_list default null),
```

### Description

Overlays an image watermark onto a source image stored in a BFILE and writes it to a destination BLOB.

### Parameters

**imageBfile**
The source image data represented as a BFILE.

**added_image**
The watermark image stored in a BFILE to be added to the source image.

**dest**
The destination BLOB for the watermarked image.

**logging**
A string that contains information about any unexpected behavior that occurred during the watermarking operation. If the operation is successful, an empty string is returned.

**watermark_properties**
A string list of name-value pairs that define attributes of the watermark image, including: width, height, position, position_x, position_y, and transparency. See Table D–2 for the complete list of watermark properties.

### Usage Notes

Calling this method processes the image into the destination BLOB from any source BFILE.

See Section D.6 for more information about watermarking operations and watermark properties.

### Pragmas

None.

### Exceptions

ORDImageExceptions.NULL_CONTENT

This exception is raised when the source image or added image is NULL.

ORDImageExceptions.NULL_DESTINATION

This exception is raised when the destination image BLOB is NULL.

See Appendix G for more information about these exceptions.

## Examples

Add a watermark image to an image BFILE:

```
-- add image as watermark for BFILE
DECLARE
  source_image   BFILE := BFILENAME('IMAGEDIR','testimg.jpg');
  added_image   BFILE := BFILENAME('IMAGEDIR','testlogo.png');
  dest_image BLOB;
  prop ordsys.ord_str_list;
  logging VARCHAR2(2000);
BEGIN
  SELECT img INTO dest_image FROM timg WHERE N = 4003 FOR UPDATE;

  -- specify properties
  prop := ordsys.ord_str_list(
                  'position=bottomright',
                  'transparency=0.2');

  -- add image watermark to source image
  ORDSYS.ORDImage.applyWatermark(source_image, added_image, dest_image, logging,
prop);

  UPDATE timg SET img = dest_image WHERE N = 4003;
  COMMIT;
EXCEPTION
   WHEN OTHERS THEN
   RAISE;
END;
/
```

where:

- `testimg.jpg`: the source image (in JPEG format) stored in the table. The source image is larger than the watermark image.
- `testlogo.png`: the watermark image (in PNG format) stored in the table. The watermark image is overlaid onto the bottom right position of the source image.
- `4003`: the resulting watermarked image stored in the table.

## applyWatermark( ) image for BLOBs

### Format

```
applyWatermark(imageBlob     IN BLOB,
                added_image      IN BLOB,
                 dest         IN OUT NOCOPY BLOB,
                 logging OUT VARCHAR2,
                 watermark_properties IN ordsys.ord_str_list default null),
```

### Description

Overlays an image watermark onto a source image stored in a BLOB and writes it to a destination BLOB.

### Parameters

**imageBlob**
The source image data represented as a BLOB.

**added_image**
The watermark image stored in a BLOB to be added to the source image.

**dest**
The destination BLOB for the watermarked image.

**logging**
A string that contains information about any unexpected behavior that occurred during the watermarking operation. If the operation is successful, an empty string is returned.

**watermark_properties**
A string list of name-value pairs that define attributes of the watermark image, including: width, height, position, position_x, position_y, and transparency. See Table D–2 for the complete list of watermark properties.

### Usage Notes

Because temporary LOBs do not have read consistency, you cannot use the same temporary LOB for both the imageBlob and dest parameters.

Calling this method processes the image into the destination BLOB from any source BLOB.

See Section D.6 for more information about watermarking operations and watermark properties.

### Pragmas

None.

### Exceptions

ORDImageExceptions.NULL_CONTENT

This exception is raised when the source image or added image is NULL.

ORDImageExceptions.NULL_DESTINATION

This exception is raised when the destination image BLOB is NULL.

See Appendix G for more information about this exception.

## Examples

Add a watermark image to an image BLOB:

```
-- add image as watermark for BLOB
DECLARE
  source_image      BLOB;
  added_image       BLOB;
  dest_image        BLOB;
  prop ordsys.ord_str_list;
  logging VARCHAR2(2000);
BEGIN
  SELECT img INTO source_image FROM timg WHERE N = 4001;
  SELECT img INTO added_image FROM timg WHERE N = 4002;
  SELECT img INTO dest_image FROM timg WHERE N = 4003 FOR UPDATE;

  -- specify properties
  prop := ordsys.ord_str_list(
                  'position=bottomright',
                  'transparency=0.2');

  -- add image watermark to source image
  ORDSYS.ORDImage.applyWatermark(source_image, added_image, dest_image, logging,
prop);

  UPDATE timg SET img = dest_image WHERE N = 4003;
  COMMIT;
EXCEPTION
   WHEN OTHERS THEN
   RAISE;
END;
/
```

where:

- `4001`: the source image (in JPEG format) stored in the table. The source image is larger than the watermark image.
- `4002`: the watermark image (in PNG format) stored in the table. The watermark image is overlaid onto the bottom right position of the source image.
- `4003`: the resulting watermarked image stored in the table.

## applyWatermark( ) text for BFILEs

### Format

```
applyWatermark(imageBfile    IN OUT NOCOPY BFILE,
               added_text      IN VARCHAR2,
               dest          IN OUT NOCOPY BLOB,
               logging OUT VARCHAR2,
               watermark_properties IN ordsys.ord_str_list default null),
```

### Description

Overlays a text watermark onto a source image stored in a BFILE and writes it to a destination BLOB.

### Parameters

**imageBfile**
The source image data represented as a BFILE.

**added_text**
The watermark text stored in a string to be added to the source image.

**dest**
The destination BLOB for the watermarked image.

**logging**
A string that contains information about any unexpected behavior that occurred during the watermarking operation. If the operation is successful, an empty string is returned. Otherwise, this method returns a string that describes the unexpected behavior. For example: if watermark text is so long that it is truncated, this string is returned: `WARNING: text is too long and truncated.`

**watermark_properties**
A string list of name-value pairs that define attributes of the watermark text, including: font_name, font_style, font_size, text_color, position_x, position_y, and transparency. See Table D–2 for the complete list of watermark properties.

### Usage Notes

Calling this method processes the image into the destination BLOB from any source BFILE.

See Section D.6 for more information about watermarking operations and watermark properties.

### Pragmas

None.

### Exceptions

ORDImageExceptions.NULL_CONTENT

This exception is raised when the source image is NULL.

ORDImageExceptions.NULL_DESTINATION

This exception is raised when the destination image BLOB is NULL.

See Appendix G for more information about these exceptions.

## Examples

Add watermark text to an image BFILE:

```
-- add text as watermark for BFILE
DECLARE
  source_image   BFILE := BFILENAME('IMAGEDIR','testimg.jpg');
  added_text   varchar2(200);
  dest_image BLOB;
  prop ordsys.ord_str_list;
  logging VARCHAR2(2000);
BEGIN
  SELECT img INTO dest_image FROM timg WHERE N = 4003 FOR UPDATE;

  added_text := 'Oracle Multimedia © 2009';
  -- specify properties
  prop := ordsys.ord_str_list(
                  'font_name=Times New Roman',
                  'font_style=bold',
                  'font_size=50',
                  'text_color=red',
                  'position_x=100',
                  'position_y=100',
                  'transparency=0.6');

  -- add text watermark to source image
  ORDSYS.ORDImage.applyWatermark(source_image, added_text, dest_image, logging,
prop);

  UPDATE timg SET img = dest_image WHERE N = 4003;
  COMMIT;
EXCEPTION
   WHEN OTHERS THEN
   RAISE;
END;
/
```

where:

- `testimg.jpg`: the source image (in JPEG format) stored in the table.
- `4003`: the resulting watermarked image stored in the table.

## applyWatermark( ) text for BLOBs

### Format

```
applyWatermark(imageBlob     IN BLOB,
               added_text       IN VARCHAR2,
               dest          IN OUT NOCOPY BLOB,
               logging OUT VARCHAR2,
               watermark_properties IN ordsys.ord_str_list default null),
```

### Description

Overlays a text watermark onto a source image stored in a BLOB and writes it to a destination BLOB.

### Parameters

**imageBfile**
The source image data represented as a BLOB.

**added_text**
The watermark text stored in a string to be added to the source image.

**dest**
The destination BLOB for the watermarked image.

**logging**
A string that contains information about any unexpected behavior that occurred during the watermarking operation. If the operation is successful, an empty string is returned. Otherwise, this method returns a string that describes the unexpected behavior. For example: if watermark text is so long that it is truncated, this string is returned: `WARNING: text is too long and truncated.`

**watermark_properties**
A string list of name-value pairs that define attributes of the watermark text, including: font_name, font_style, font_size, text_color, position_x, position_y, and transparency. See Table D–2 for the complete list of watermark properties.

### Usage Notes

Because temporary LOBs do not have read consistency, you cannot use the same temporary LOB for both the imageBlob and dest parameters.

Calling this method processes the image into the destination BLOB from any source BLOB.

See Section D.6 for more information about watermarking operations and watermark properties.

### Pragmas

None.

### Exceptions

ORDImageExceptions.NULL_CONTENT

This exception is raised when the source image is NULL.

ORDImageExceptions.NULL_DESTINATION

This exception is raised when the destination image BLOB is NULL.

See Appendix G for more information about this exception.

## Examples

Add watermark text to an image BLOB:

```
-- add text as watermark for BLOB
DECLARE
  source_image      BLOB;
  added_text  varchar2(200);
  dest_image        BLOB;
  prop ordsys.ord_str_list;
  logging VARCHAR2(2000);
BEGIN
  SELECT img INTO source_image FROM timg WHERE N = 4001;
  SELECT img INTO dest_image FROM timg WHERE N = 4003 FOR UPDATE;

  added_text := 'Oracle Multimedia © 2009';
  -- specify properties
  prop := ordsys.ord_str_list(
                  'font_name=Times New Roman',
                  'font_style=bold',
                  'font_size=50',
                  'text_color=red',
                  'position_x=100',
                  'position_y=100',
                  'transparency=0.6');

  -- add text watermark to source image
  ORDSYS.ORDImage.applyWatermark(source_image, added_text, dest_image, logging,
prop);

  UPDATE timg SET img = dest_image WHERE N = 4003;
  COMMIT;
EXCEPTION
   WHEN OTHERS THEN
   RAISE;
END;
/
```

where:

- `4001`: the source image (in JPEG format) stored in the table.
- `4003`: the resulting watermarked image stored in the table.

## getMetadata( ) for BFILEs

### Format

```
getMetadata(imageBfile      IN NOCOPY BFILE,
            metadataType    IN VARCHAR2 DEFAULT 'ALL'
RETURN XMLSequenceType;
```

### Description

Extracts the specified types of metadata from the imageBfile and returns an array of schema-valid XML documents. If no matching metadata is found, an empty array is returned.

### Parameters

**imageBfile**
The image data represented as a BFILE.

**metadataType**
A string that identifies the types of embedded metadata to extract. Valid values are: `ALL`, `ORDIMAGE`, `XMP`, `EXIF`, and `IPTC-IIM`. The default value is `ALL`.

### Usage Notes

When the value of input parameter metadataType is `ALL`, and two or more types of supported metadata are present in the image, this method returns several XML documents, one for each type of metadata found. For other values of the input parameter, the method returns zero or one XML document.

Each document is stored as an instance of XMLType, and is based on one of the metadata schemas. Use the XQuery function `fn:namespace-uri` to determine the type of metadata represented in that document.

See Appendix F for a description of the supported metadata schemas.

**See Also:**

- *Oracle Multimedia User's Guide* for more information about the metadata feature
- *Oracle XML DB Developer's Guide* for more information about XQuery functions

### Pragmas

None.

### Exceptions

ORDImageExceptions.NULL_CONTENT

This exception is raised when the imageBfile parameter is NULL.

See Appendix G for more information about this exception.

## Examples

Extract the embedded metadata from an imageBfile, and return an array of schema-valid XML documents:

```
DECLARE
  imageBfile  BFILE := BFILENAME('MEDIA_DIR','keyboard.jpg');
  metav       XMLSequenceType;
  ns          varchar2(4000);
BEGIN

  metav := ORDSYS.ORDImage.getMetadata(imageBfile, 'ALL');

  -- print the namespace of each metadata document
  FOR i in 1..metav.count LOOP
    select xmlcast(xmlquery('fn:namespace-uri($x)'
          passing metav(i) as "x" returning content) as varchar2(4000))
      into ns from dual;
    DBMS_OUTPUT.PUT_LINE('namespace: ' || ns);
  END LOOP;

  EXCEPTION
  WHEN ORDSYS.ORDImageExceptions.NULL_CONTENT THEN
    DBMS_OUTPUT.PUT_LINE('imageBlob is null');
  WHEN OTHERS THEN
    RAISE;
END;
/
```

## getMetadata( ) for BLOBs

### Format

```
getMetadata(imageBlob      IN NOCOPY BLOB,
            metadataType   IN VARCHAR2 DEFAULT 'ALL')
RETURN XMLSequenceType;
```

### Description

Extracts the specified types of metadata from the imageBlob and returns an array of schema-valid XML documents. If no matching metadata is found, an empty array is returned.

### Parameters

**imageBlob**
The image data represented as a BLOB.

**metadataType**
A string that identifies the types of embedded metadata to extract. Valid values are: `ALL`, `ORDIMAGE`, `XMP`, `EXIF`, and `IPTC-IIM`. The default value is `ALL`.

### Usage Notes

When the value of input parameter metadataType is `ALL`, and two or more types of supported metadata are present in the image, this method returns several XML documents, one for each type of metadata found. For other values of the input parameter, the method returns zero or one XML document.

Each document is stored as an instance of XMLType, and is based on one of the metadata schemas. Use the XQuery function `fn:namespace-uri` to determine the type of metadata represented in that document.

See Appendix F for a description of the supported metadata schemas.

**See Also:**

- *Oracle Multimedia User's Guide* for more information about the metadata feature
- *Oracle XML DB Developer's Guide* for more information about XQuery functions

### Pragmas

None.

### Exceptions

ORDImageExceptions.NULL_CONTENT

This exception is raised when the imageBlob parameter is NULL.

See Appendix G for more information about this exception.

## Examples

Extract the embedded metadata from an imageBlob, and return an array of schema-valid XML documents:

```
DECLARE
  imageBlob BLOB;
  metav     XMLSequenceType;
  ns        varchar2(4000);
BEGIN
  SELECT ad_photo
  INTO imageBlob
  FROM pm.print_media
  WHERE product_id = 3106;

  metav := ORDSYS.ORDImage.getMetadata(imageBlob, 'ALL');

  -- print the namespace of each metadata document
  FOR i in 1..metav.count LOOP
    select xmlcast(xmlquery('fn:namespace-uri($x)'
          passing metav(i) as "x" returning content) as varchar2(4000))
      into ns from dual;
    DBMS_OUTPUT.PUT_LINE('namespace: ' || ns);
  END LOOP;

  EXCEPTION
  WHEN ORDSYS.ORDImageExceptions.NULL_CONTENT THEN
    DBMS_OUTPUT.PUT_LINE('imageBlob is null');
  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.

See Appendix G for more information about this exception.

### 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 these 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 on).

**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.

See Appendix G for more information about this exception.

## 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;
/
```

# 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.

See Appendix G for more information about this exception.

## 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 these 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 on).

**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.

See Appendix G for more information about this exception.

## 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;
```

/

# 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 5–1. Table 5–2 shows additional changes that can be made only to raw pixel and foreign images.

See Appendix D for more information about process( ) operators.

The process( ) method changes image attributes, therefore if you are storing image attributes, 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.

See Appendix G for more information about this exception.

## 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:** Changing the length on one axis (for example, xScale=2.0) does not affect the length on the other axis; thus, it distorts the image. Only the xScale and yScale operators can be combined in a single scale operation; any other combination of scale operators causes 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 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 5–1 and Table 5–2 for information about image processing operators.

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, call the getProperties( ) method on the destination image after calling the processCopy( ) method.

See Appendix D for more information about 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.

See Appendix G for more information about these exceptions.

## 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;
/
```

# 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 5–1 and Table 5–2 for information about image processing operators.

Because temporary LOBs do not have read consistency, you cannot use the same temporary LOB for both the imageBlob and dest parameters.

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, call the getProperties( ) method on the destination image after calling the processCopy( ) method.

See Appendix D for more information about 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.

See Appendix G for more information about this exception.

## 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;
/
```

# putMetadata( ) for BFILEs

## Format

```
putMetadata(imageBfile      IN NOCOPY BFILE,
            dest            IN OUT NOCOPY BLOB
            xmlData         IN NOCOPY XMLType,
            metadataType    IN VARCHAR2 DEFAULT 'XMP',
            encoding        IN VARCHAR2 DEFAULT "UTF-8");
```

## Description

Accepts a BFILE containing an image and a schema-valid XML document, and creates a binary packet suitable for embedding in the target image file format. The packet is encoded according to the value of the encoding parameter. If the value of the metadataType parameter is `XMP`, this method writes a new XMP packet to the image, replacing any existing XMP packets. The new image file with embedded metadata is returned in the dest parameter.

## Parameters

**imageBfile**
The BFILE handle to the image.

**dest**
The BLOB to receive the image containing the embedded metadata.

**xmlData**
The XMLtype that contains a schema-valid XML document for the indicated metadataType. If the value of the metadataType parameter is `XMP`, the root element should contain a well-formed RDF document.

**metadataType**
A string that specifies the type of metadata to write. The valid value is `XMP`; it is also the default.

**encoding**
The character encoding to be used in the image file. Valid values are: `UTF-8`, `UTF-16`, `UTF-16BE`, and `UTF-16LE`. The default is `UTF-8`.

## Usage Notes

The binary metadata packet generated from the same xmlData input may have different sizes for different encodings. Different image file formats support different encodings, and may restrict the binary metadata packet size. The restrictions of the supported image formats are as follows:

- GIF89a supports UTF-8 encoding only.
- JPEG requires a binary packet size of less than 65502 bytes.
- TIFF requires a binary packet size of less than 4 gigabytes.

**See Also:**

*Oracle Multimedia User's Guide* for more information about the metadata feature

## Pragmas

None.

## Exceptions

ORDImageExceptions.NULL_CONTENT

This exception is raised when the image is NULL.

ORDImageExceptions.NULL_DESTINATION

This exception is raised when the destination image is NULL.

See Appendix G for more information about these exceptions.

## Examples

Insert a new image into the table `pm.print_media`. The new image is a copy of the image `keyboard.jpg` from the `MEDIA_DIR` directory object, with updated XMP metadata:

```
DECLARE
  imageBfile        BFILE := BFILENAME('MEDIA_DIR','keyboard.jpg');
  dest              BLOB;
  xmlData           XMLType;
BEGIN
  SELECT ad_photo
  INTO dest
  FROM pm.print_media
  WHERE product_id = 3106 FOR UPDATE;


  xmlData := xmltype(
          '<xmpMetadata xmlns="http://xmlns.oracle.com/ord/meta/xmp">' ||
          '<rdf:RDF xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"' ||
          '  xmlns:dc="http://purl.org/dc/elements/1.1/">' ||
          '<rdf:Description>' ||
          '  <dc:rights>' ||
          '    <rdf:Alt>' ||
          '      <rdf:li xml:lang="en-us">' ||
          '        Oracle Corporation' ||
          '      </rdf:li>' ||
          '    </rdf:Alt>'||
          '  </dc:rights>' ||
          '</rdf:Description>' ||
          '</rdf:RDF>' ||
          '</xmpMetadata>', 'http://xmlns.oracle.com/ord/meta/xmp');


  ORDSYS.ORDImage.putMetadata(imageBfile, dest, xmlData, 'xmp', 'utf-8');


  UPDATE pm.print_media
  SET ad_photo = dest
  WHERE product_id = 3106;
  COMMIT;
```

```
  EXCEPTION
    WHEN ORDSYS.ORDImageExceptions.NULL_CONTENT THEN
      DBMS_OUTPUT.PUT_LINE('image source is null');
    WHEN ORDSYS.ORDImageExceptions.NULL_DESTINATION THEN
      DBMS_OUTPUT.PUT_LINE('image destionation is null');
    WHEN OTHERS THEN
    RAISE;
END;
/
```

## putMetadata( ) for BLOBs

### Format

```
putMetadata(imageBlob      IN NOCOPY BLOB,
            dest           IN OUT NOCOPY BLOB
            xmlData        IN NOCOPY XMLType,
            metadataType   IN VARCHAR2 DEFAULT 'XMP',
            encoding       IN VARCHAR2 DEFAULT "UTF-8");
```

### Description

Accepts a BLOB containing an image and a schema-valid XML document, and creates a binary packet suitable for embedding in the target image file format. The packet is encoded according to the value of the encoding parameter. If the value of the metadataType parameter is `XMP`, this method writes a new XMP packet to the image, replacing any existing XMP packets. The new image file with embedded metadata is returned in the dest parameter.

### Parameters

**imageBlob**
The BLOB handle to the image.

**dest**
The BLOB to receive the image containing the embedded metadata.

**xmlData**
The XMLtype that contains a schema-valid XML document for the indicated metadataType. If the value of the metadataType parameter is `XMP`, the root element should contain a well-formed RDF document.

**metadataType**
A string that specifies the type of metadata to write. The valid value is `XMP`; it is also the default.

**encoding**
The character encoding to be used in the image file. Valid values are: `UTF-8`, `UTF-16`, `UTF-16BE`, and `UTF-16LE`. The default is `UTF-8`.

### Usage Notes

Because temporary LOBs do not have read consistency, you cannot use one temporary LOB for both the imageBlob and dest parameters. The binary metadata packet generated from the same xmlData input may have different sizes for different encodings. Different image file formats support different encodings, and may restrict the binary metadata packet size. The restrictions of the supported image formats are as follows:

- GIF89a supports UTF-8 encoding only.
- JPEG requires a binary packet size of less than 65502 bytes.
- TIFF requires a binary packet size of less than 4 gigabytes.

**See Also:**

*Oracle Multimedia User's Guide* for more information about the metadata feature

## Pragmas

None.

## Exceptions

ORDImageExceptions.NULL_CONTENT

This exception is raised when the image is NULL.

ORDImageExceptions.NULL_DESTINATION

This exception is raised when the destination image is NULL.

See Appendix G for more information about these exceptions.

## Examples

Replace an image in the table `pm.print_media` with updated XMP metadata:

```
DECLARE
  imageBlob        BLOB;
  tmp              BLOB;
  xmlData          XMLType;
  ctx              RAW(64):=NULL;
BEGIN
  SELECT ad_photo
  INTO imageBlob
  FROM pm.print_media
  WHERE product_id = 3106 FOR UPDATE;


  xmlData := xmltype(
          '<xmpMetadata xmlns="http://xmlns.oracle.com/ord/meta/xmp">' ||
          '<rdf:RDF xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"' ||
          '  xmlns:dc="http://purl.org/dc/elements/1.1/">' ||
          '<rdf:Description>' ||
          '  <dc:rights>' ||
          '    <rdf:Alt>' ||
          '      <rdf:li xml:lang="en-us">' ||
          '        Oracle Corporation' ||
          '      </rdf:li>' ||
          '    </rdf:Alt>'||
          '  </dc:rights>' ||
          '</rdf:Description>' ||
          '</rdf:RDF>' ||
          '</xmpMetadata>', 'http://xmlns.oracle.com/ord/meta/xmp');


  tmp := imageBlob;
  ORDSYS.ORDImage.putMetadata(tmp, imageBlob, xmlData, 'xmp', 'utf-8');


  UPDATE pm.print_media
  SET ad_photo = imageBlob
  WHERE product_id = 3106;
  COMMIT;
```

```
  EXCEPTION
    WHEN ORDSYS.ORDImageExceptions.NULL_CONTENT THEN
      DBMS_OUTPUT.PUT_LINE('image source is null');
    WHEN ORDSYS.ORDImageExceptions.NULL_DESTINATION THEN
      DBMS_OUTPUT.PUT_LINE('image destionation is null');
    WHEN OTHERS THEN
      RAISE;
END;
/
```

# Static Methods Unique to the ORDVideo Object Type Relational Interface

This section presents reference information about these Oracle Multimedia static methods, which are unique to the ORDVideo relational interface:



The relational interface adds Oracle Multimedia support to video data stored in BLOBs and BFILEs rather than in the ORDVideo object type. The static methods that are unique to the ORDVideo relational interface are defined in the `ordvspec.sql` file.

## 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. (See Section 7.3.)

**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

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`.

ORDVideoExceptions.VIDEO_PLUGIN_EXCEPTION

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

See Appendix G for more information about these exceptions.

### 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 these 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. (See Section 7.3.)

**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

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`.

ORDVideoExceptions.VIDEO_PLUGIN_EXCEPTION

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

See Appendix G for more information about these exceptions.

## 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;
/
```

# 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. (See Section 7.3.)

**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

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`.

ORDVideoExceptions.VIDEO_PLUGIN_EXCEPTION

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

See Appendix G for more information about these exceptions.

## 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 these 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. (See Section 7.3.)

**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

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`.

ORDVideoExceptions.VIDEO_PLUGIN_EXCEPTION

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

See Appendix G for more information about these exceptions.

## 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;
/
```

# 8

# Oracle Multimedia ORDSource Object Type

Oracle Multimedia provides the ORDSource object type, which 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.

The ORDSource object type is defined in the `ordsrcsp.sql` file. After installation, this file is available in the Oracle home directory at:

*<ORACLE_HOME>*`/ord/im/admin` (on Linux and UNIX)

*<ORACLE_HOME>*`\ord\im\admin` (on Windows)

This chapter contains these sections:



> **Note:** This object is used only by other Oracle Multimedia objects. The information in this chapter is included for reference only. Oracle does not recommend that you use this type.

## 8.1 Important Notes for ORDSource Methods

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 must 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 must 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 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. The attributes for this object type are defined as follows in the `ordsrcsp.sql` file:

```
------------------
-- TYPE ATTRIBUTES
------------------
localData           BLOB,
srcType             VARCHAR2(4000),
srcLocation         VARCHAR2(4000),
srcName             VARCHAR2(4000),
updateTime          DATE,
local               NUMBER,
```

where:

- localData: the locally stored multimedia data stored as a BLOB within the object. Depending on the block size, up to 8 terabytes (TB) to 128 TB 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. (See Table 2–1 for the list of valid values.)
- srcLocation: the place where data can be found based on the srcType value. (See Table 2–2 for the list of valid values.)
- srcName: the data object name. (See Table 2–3 for the list of valid values.)
- updateTime: the time at which the data was last updated.
- local: a flag that indicates whether the data is local. The valid values are:
  - `1`: the data is in the BLOB.
  - `0`: the data is in external sources.
  - `NULL`: the data is local. This value may indicate a default state when you first insert an empty row.

**See Also:**

*Oracle Database SecureFiles and Large Objects Developer's Guide* for more information about using BLOBs

# ORDSource Methods

This section presents ORDSource reference information about these ORDSource methods, which are provided for source data manipulation:



**See Also:**

*Oracle Database Concepts* for more information about object types and methods

## 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. (See Section 8.1.)

## 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 on.

## Pragmas

None.

## Exceptions

ORDSourceExceptions.INCOMPLETE_SOURCE_INFORMATION

This exception is raised if you call the close( ) method and the value of 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.

See Appendix G for more information about these exceptions.

## 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 provides native support only for a source.srcType value of `FILE`. In this case, the data is exported to a file in a directory that is accessible to Oracle Database. User-defined sources can support the export( ) method to provide WRITE access to other types of data stores.

## Parameters

**ctx**
The source plug-in context information. (See Section 8.1.)

**source_type**
The type of the external source data. This parameter is not case-sensitive. (See Table 2–1.)

**source_location**
The location to which the source data is to be exported. (See Table 2–2.)

**source_name**
The name of the object to which the source data is to be exported. (See Table 2–3.)

## Usage Notes

This method exports data from the localData attribute to the external data source specified by the input parameters.

Calling this method uses the ORDPLUGINS.ORDX_<srcType>_SOURCE plug-in package.

After exporting data, all attributes remain unchanged except the srcType, srcLocation, and srcName attributes, which are updated with input parameter values. After calling the export( ) method, you can call the clearLocal( ) method to indicate that the data is stored outside the database, and then call the deleteLocalContent( ) method to delete the content of the local data in the localData attribute.

When the source_type parameter has a value of `FILE`, the source_location parameter specifies the name of an Oracle directory object, and the source_name parameter specifies the name of the file in which the data is to be contained.

The export( ) method writes only to a database directory object that the user has privilege to access. That is, you can access a directory object that you have created

using the SQL statement CREATE DIRECTORY, or one to which you have been granted READ and WRITE access.

For example, the following SQL*Plus commands create a directory object and grant the user `ron` permission to read and write to any file within the directory `c:\mydir\work`:

```
CONNECT sys as sysdba
Enter password: password
CREATE OR REPLACE DIRECTORY FILE_DIR AS 'c:\mydir\work';
GRANT READ,WRITE ON DIRECTORY FILE_DIR TO ron;
```

Now, the user `ron` can export an image to the `testimg.jpg` file in this directory using the export( ) method of the ORDImage object:

```
img.export('FILE', 'FILE_DIR', testimg.jpg');
```

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.

See Appendix G for more information about these exceptions.

## Examples

None.

# getBFile( )

## Format

getBFile( ) RETURN BFILE;

## Description

Returns a BFILE handle, if the value of the srcType attribute 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`.

See Appendix G for more information about these exceptions.

## 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 is to be allocated and initialized as a part of this call.

### Parameters

**ctx**
The source plug-in context information. (See Section 8.1.)

**tempLob**
An uninitialized BLOB locator, which is to 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:

`DBMS_LOB.CALL`

`DBMS_LOB.TRANSACTION`

`DBMS_LOB.SESSION`

**cache**
A Boolean value that indicates whether 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.)

See Appendix G for more information about these exceptions.

## 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. (See Section 8.1.)

### Usage Notes

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 getContentLength( ) method and the value of the srcType attribute is `NULL`.

See Appendix G for more information about these exceptions.

### 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. (See Section 8.1.)

**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 must 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 Fusion Middleware.

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 getSourceAddress( ) method and the value of the srcType attribute is `NULL`.

See Appendix G for more information about these exceptions.

### 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`.

See Appendix G for more information about these exceptions.

### 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`.

See Appendix G for more information about these exceptions.

## 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 time stamp 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. (See Section 8.1.)

**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

Before calling this method, call setSourceInformation( ) method to set the srcType, srcLocation, and srcName attribute values to describe where the data source is located.

Calling this method uses the ORDPLUGINS.ORDX_<srcType>_SOURCE plug-in package.

After importing data from an external 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).

If the value of the srcType attribute is `FILE`, the srcLocation attribute contains the name of a database directory object which contains the file to be imported, and the srcName attribute contains the name of the file to be imported. You must ensure that the directory for the external source location exists or is created before you use this method.

The import( ) method reads only from a database directory object that the user has privilege to access. That is, you can access a directory object that you have created using the SQL statement CREATE DIRECTORY, or one to which you have been granted READ access.

For example, the following SQL*Plus commands create a directory object and grant the user `ron` permission to read any file within the directory `c:\mydir\work`:

```
CONNECT sys as sysdba
Enter password: password
CREATE OR REPLACE DIRECTORY FILE_DIR AS 'c:\mydir\work';
GRANT READ ON DIRECTORY FILE_DIR TO ron;
```

If the value of the srcType attribute is HTTP, the srcLocation attribute contains the base URL needed to find the source directory that contains the object to be imported, and the srcName attribute contains the name of the object to be imported.

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 Linux and UNIX, setting the environment variable http_proxy to a URL specifies that the UTL_HTTP package must 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 not be used for URLs in the specified domain.

**See Also:**

*Oracle Database PL/SQL Packages and Types Reference* for more information about the UTL_HTTP package

If the value of the source.srcType attribute is a user-defined name, the source.srcLocation attribute contains an identifier string required to access the user-defined object to be imported, and the source.srcName attribute contains the name of the object to be imported.

## 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.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.NULL_SOURCE

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

See Appendix G for more information about these exceptions.

## 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 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. (See Section 8.1.)

**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. (See Table 2–1.)

**source_location**
The location from which the source data is to be imported. This also sets the srcLocation attribute. (See Table 2–2.)

**source_name**
The name of the source data to be imported. This also sets the srcName attribute. (See Table 2–3.)

## 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( ) method followed by an import( ) method.

Calling this method uses the ORDPLUGINS.ORDX_<srcType>_SOURCE plug-in package.

If the value of the source_type parameter is `FILE`, the source_location parameter contains the name of a database directory object that contains the file to be imported,

and the source_name parameter contains the name of the file to be imported. You must ensure that the directory indicated by the source_location parameter exists or is created before you use this method.

The importFrom( ) method reads only from a database directory object that the user has privilege to access. That is, you can access a directory object that you have created using the SQL statement CREATE DIRECTORY, or one to which you have been granted READ access.

For example, the following SQL*Plus commands create a directory object and grant the user `ron` permission to read any file within the directory `c:\mydir\work`:

```
CONNECT sys as sysdba
Enter password: password
CREATE OR REPLACE DIRECTORY FILE_DIR AS 'c:\mydir\work';
GRANT READ ON DIRECTORY FILE_DIR TO ron;
```

If the value of the source_type parameter is `HTTP`, the source_location parameter contains the base URL needed to find the source directory that contains the object to be imported, and the source_name parameter contains the name of the object to be imported.

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 Linux and UNIX, setting the environment variable http_proxy to a URL specifies that the UTL_HTTP package must 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 not be used for URLs in the specified domain.

**See Also:**

*Oracle Database PL/SQL Packages and Types Reference* for more information about the UTL_HTTP package

If the value of the source_type parameter is a user-defined name, the source_location parameter contains an identifier string required to access the user-defined object to be imported, and the source_name parameter contains the name of the object to be imported.

## 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.NULL_SOURCE

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

See Appendix G for more information about these exceptions.

## 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. (See Section 8.1.)

## 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 on.

## Pragmas

None.

## Exceptions

ORDSourceExceptions.INCOMPLETE_SOURCE_INFORMATION

This exception is raised if you call the open( ) method and the value of 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.

See Appendix G for more information about these exceptions.

## 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. (See Section 8.1.)

**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.

See Appendix G for more information about these exceptions.

### 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. (See Section 8.1.)

**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 is to 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. 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.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.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`.

See Appendix G for more information about these exceptions.

## 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 Table 2–1.)

**source_location**
The location of the external source data. (See Table 2–2.)

**source_name**
The name of the external source data. See Table 2–3.)

### 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 of the source_type parameter is `NULL`.

See Appendix G for more information about this exception.

### 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. (See Section 8.1.)

**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 on.

### Pragmas

None.

### Exceptions

ORDSourceExceptions.INCOMPLETE_SOURCE_INFORMATION

This exception is raised if you call the trim( ) method and the value of 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.

See Appendix G for more information about these exceptions.

### 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. (See Section 8.1.)

**startPos**
The start position in the source to where the buffer is to 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.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.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`.

See Appendix G for more information about these exceptions.

## Examples

None.

# A

# Audio File and Compression Formats for Use With Oracle Multimedia

This appendix describes the audio file and compression formats and other audio features supported by Oracle Multimedia, in these sections:



Find the audio data format you are interested in, and then determine the supported formats. For example, Section A.2 shows that Oracle Multimedia supports AIFF format for single channel, stereo, 8-bit and 16-bit samples, linear PCM encoding, and uncompressed format.

## A.1 Supported 3GP Data Format

The supported 3GP format ID, file format, file extension, and MIME type are as follows:

- Format ID: 3GP
- File Format: 3GP
- File extension: .3gp
- MIME type: audio/3gpp

## A.2 Supported AIFF Data Formats

The supported AIFF format ID, file format, file extension, MIME type, audio features, compression format, and encoding/compression type are as follows:

- 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.3 Supported AIFF-C Data Formats

The supported AIFF-C format ID, file format, file extension, MIME type, and audio features are as follows:

- 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.4 Supported AU Data Formats

The supported AU format ID, file format, file extension, MIME type, and audio features are as follows:

- 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 |
| 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 |

*Table A–2 (Cont.) AU Data Compression Formats and Types*

| Compression Format | Encoding/Compression Types |
|---|---|
| ADPCM G723_5 | ADPCM_G723_5 |
| 8-bit a-law samples | ALAW |

## A.5 Supported Audio MPEG Data Formats

The supported audio MPEG formats are MPEG1, MPEG2, and MPEG4, as described in the following sections:

- Supported MPEG1 and MPEG2 Data Formats
- Supported MPEG4 Data Formats

### A.5.1 Supported MPEG1 and MPEG2 Data Formats

The MPEG1 and MPEG2 format ID, file format, file extension, MIME type, and audio features are as follows:

- 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 |

The following subsection provides more detail about MPEG1 and MPEG2 audio formats:

- Supported ID3 Audio File Data Tagging Format

#### A.5.1.1 Supported ID3 Audio File Data Tagging Format

Within the MPEG1 and MPEG2 audio formats, for Layer I, Layer II, and Layer III, these ID3V1.1 tags are supported:

- Song title
- Artist
- Album

- Year
- Comment
- Genre

The metadata information in these tags is extracted into the comments attribute of the ORDAudio object. See ORDAudio Object Type on page 3-3 for more information about ORDAudio object type attributes.

### A.5.2 Supported MPEG4 Data Formats

The MPEG4 format ID, file format, file extension, and MIME type features are as follows:

- Format ID: MP4
- File format: MP4
- File extension: .mp4
- MIME type: audio/mp4

## A.6 Supported RealNetworks Real Audio Data Format

The supported RealNetworks Real Audio format ID, file format, file extensions, MIME type, and audio features are as follows:

- 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.7 Supported WAV Data Formats

The supported WAV format ID, file format, file extension, MIME type, and audio features are as follows:

- 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 |
| 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 |

*Table A–4 (Cont.) WAV Data Compression Formats and Types*

| Compression Formats | Encoding/Compression Types |
|---|---|
| Olivetti GSM Wave Format | OLIGSM |
| Olivetti ADPCM Wave Format | OLIADPCM |
| Olivetti CELP Wave Format | OLICELP |
| Olivetti SBC Wave Format | OLISBC |
| Olivetti OPR Wave Format | OLIOPR |

## A.8 Supported Windows Media File Format (ASF) for Audio Data

Oracle Multimedia supports the Advanced Systems Format (ASF) file format used by Microsoft Windows Media. These files are denoted by the suffix .wma. The Oracle Multimedia objects ORDAudio and ORDDoc can recognize ASF format data and extract attributes about the audio media from the data.

---

**Note:** To access these formats, install the software from the Oracle Multimedia Software section of the Oracle Technology Network Web site at

http://www.oracle.com/technology/products/multimedia/

See *Support for the Windows Media File Format (ASF)*.

---

**See Also:**

http://www.microsoft.com/windows/windowsmedia/ for more information about the Advanced Systems Format used by Microsoft Windows Media

The supported Windows Media file extension and MIME type for audio data stored in an ORDAudio object are as follows:

- File extension: .wma
- MIME type: audio/x-ms-wma

Table A–5 lists the supported Windows Media audio compression format names and format codes for the ASF formats.

*Table A–5 ASF Audio Compression Formats and Format Codes*

| Compression Formats | Format Codes |
|---|---|
| Windows Media Audio | WMA |
| Windows Media Audio 9 Professional | WMA_9_PROFESSIONAL |
| Windows Media Audio 9 Voice | WMA_9_VOICE |
| Windows Media Audio Lossless | WMA_9_LOSSLESS |
| Sipro Lavs ACELP.net | SIPRO_ACEHLP |
| GSM-AMR Fixed bitrate | GSM_AMR_FIXED |

***Table A–5 (Cont.) ASF Audio Compression Formats and Format Codes***

| Compression Formats | Format Codes |
|---|---|
| GSM-AMR Variable bitrate | GSM_AMR_VARIABLE |

# B

# Image File and Compression Formats for Use With Oracle Multimedia

This appendix describes the image file and compression formats supported by Oracle Multimedia, in these sections:



Find the image data format you are interested in, and then determine the supported formats. For example, Section B.1 shows that Oracle Multimedia supports BMPF, the Microsoft Windows bitmap format. And, Section B.2 shows that Oracle Multimedia supports BMPRLE, the corresponding compression format. See Section B.3 for a summary of the supported formats.

See Appendix D for information about 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 Multimedia 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; Oracle Multimedia 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 Oracle Multimedia does not provide native recognition and support, but that can sometimes be read if the image data complies with the rules outlined in Section E.10, "Foreign Image Support and the Raw Pixel Format" in Appendix E.

### 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 permitting 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 Multimedia includes a simple FlashPix decoder that always selects the largest resolution plane in a FlashPix image. Lower resolutions are not accessible. Oracle Multimedia does not write FlashPix images.

### GIFF

extension: .gif

mime: image/gif

GIFF is the Oracle Multimedia 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; Oracle Multimedia 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 Multimedia 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 Oracle Multimedia 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 Multimedia 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 are written as the most appropriate variant depending on the format of the input data content.

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 Multimedia 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 Multimedia. Oracle Multimedia 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 Multimedia 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 Oracle Multimedia 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 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 Oracle Multimedia 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 on), 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 permit an RPIX header to be prepended to other image data, thus enabling the RPIX decoder to read an otherwise compliant image format. See Appendix E for more information about this format.

In addition to its support for data with 8 bits per sample, 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, Oracle Multimedia writes only 1 or 3 band RPIX images.

### RASF

extension: .ras

mime: image/x-ora-rasf

The Sun Raster image format, called RASF by Oracle Multimedia, 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 Oracle Multimedia.

### TGAF

extension: .tga

mime: image/x-ora-tgaf

The Truevision Graphics Adapter format (TGA, or TGAF to Oracle Multimedia) 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 Oracle Multimedia.

### 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 Multimedia 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. The JPEG support in TIFF provided by Oracle Multimedia is based on the revised JPEG in TIFF specification, and not the original JPEG in TIFF specification. TIFF images in either big endian format or little endian format can be read, but Oracle Multimedia always writes big endian format TIFFs.

Although the TIFF decoder in Oracle Multimedia 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. This initial page is accessed by setting "page=0" in the process command string. Oracle Multimedia 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) to transmit 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 Multimedia 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 enable 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 Multimedia provides read support for this encoding, but does not provide write support.

### FAX3

FAX3 is the Oracle Multimedia 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 Oracle Multimedia 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 Oracle Multimedia 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 enable 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 Multimedia provides read support for this encoding, but does not provide write support.

### HUFFMAN3

HUFFMAN3 is the Oracle Multimedia 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 Multimedia 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 might not notice. While JPEG can compress most data quite well, the results might 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 Multimedia provides read, but not write, support for this encoding.

### LZW

LZW is the Oracle Multimedia 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 Oracle Multimedia 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 Oracle Multimedia 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 Oracle Multimedia 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 Oracle Multimedia 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 Oracle Multimedia 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 Formats and Image Compression Formats

This section presents these summary tables:

- 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"

**Note:** See Section B.4 for definitions of the abbreviations used in these tables.

Table B–1 summarizes the I/O 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–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 | Monochrome | 32BIT CMYK | 64BIT CMYK |
| 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 | - | - | - | - | I | I |
| 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 | Monochrome | 32BIT CMYK | 64BIT CMYK |
| 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 | I | I |
| WBMP | - | - | - | - | - | - | - | - | - | - | - | - | I O | - | - |

[1] RGB + Alpha, RGB + transparency, GRAY + Alpha, GRAY + transparency.

[2] Animated GIFFs cannot 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 is PBMF, PGMF, or PPMF as appropriate.

[6] Can decode 1 or 3 bands from an *n*-band image; only 1 or 3 bands can be encoded.

[7] TIFF image file format also supports these content formats as input or I/O, as specified: Tiled data - input, Photometric interpretation - I/O, MSB - I/O, and LSB - input; Planar (BSQ) is not supported; both MSB and LSB ordered files can be decoded; decoded output is MSB.

Table B–2 summarizes the I/O support provided for process( ) and setProperties( ) methods for image file formats relative to compression format.

***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 |
| BMPF[4] | I O | - | - | I O | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - |
| CALS | - | - | - | - | - | - | - | - | - | - | - | - | I O | - | - | - | - | - | - |
| FPIX | - | I | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - |
| GIFF | - | - | - | - | - | - | - | I O | I | - | - | - | - | - | - | - | - | - | - |
| JFIF[5] | - | I O | I | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - |
| PBMF | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | I O | I O |
| PCXF | - | - | - | - | I | - | - | - | - | - | - | - | - | - | - | - | - | - | - |
| PGMF | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | I O | I O |
| PICT | - | I O | - | - | - | - | - | - | - | - | - | - | - | - | I O | - | - | - | - |
| PNGF | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | I O | I | - | - |
| PNMF | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | O | O |

***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 | LZWHDIF[2] | FAX3[3] | FAX4[3] | HUFFMAN3[3] | PACKBITS | DEFLATE | DEFLATE-ADAM7 | ASCII | RAW |
| 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 MONOCHROME2 data only.
[4] Compression is supported only for scanlineOrder=INVERSE (inverse DIB), which is the default.
[5] Supports EXIF images.

Table B–3 summarizes the I/O support provided for process( ) and setProperties( ) methods for other format-specific characteristics, such as pixel layout, channel order, pixel order, and scanline order.

***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 | - | - | - | - |

***Table B–3 (Cont.) 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 |
| 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 cannot 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 is PBMF, PGMF, or PPMF as appropriate.

[5] Can decode 1 or 3 bands from an *n*-band image; only 1 or 3 bands can be encoded.

[6] TIFF image file format also supports these content formats as input or I/O, as specified: Tiled data - input, Photometric interpretation - I/O, MSB - I/O, and LSB - input; Planar (BSQ) is not supported; both MSB and LSB ordered files can be decoded; decoded output is MSB.

## B.4 Definitions for Abbreviations Used in the Image Format Tables

These 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.

**- (the hyphen character)**
No input or output support is provided.

# C

# Video File and Compression Formats for Use With Oracle Multimedia

This appendix describes the video file and compression formats supported by Oracle Multimedia, in these sections:



Find the video data format you are interested in, and then determine the supported formats. For example, Table C–1 shows that Oracle Multimedia 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 follows:

- 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 Oracle Multimedia 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 Oracle Multimedia.

***Table C–1 Supported Apple QuickTime 3.0 Data Compression Formats***

| Compression Format Name | Compression Format Code |
|---|---|
| Cinepak | CVID |
| JPEG | JPEG |
| Uncompressed RGB | RGB |
| Uncompressed YUV422 | YUV2 |

*Table C–1 (Cont.) Supported Apple QuickTime 3.0 Data Compression Formats*

| Compression Format Name | Compression Format Code |
|---|---|
| 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 supported Microsoft Video for Windows data format, file extension, and MIME type are as follows:

- Data 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 Oracle Multimedia obtains from the compression field of the 'strf' chunk in the AVI file. The table lists only the compression format codes recognized by Oracle Multimedia.

*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 |
| Intel Indeo 5.0 | IV50 |
| Intel Indeo 5.1 | IV51 |
| Cinepak | CVID |

## C.3 RealNetworks Real Video Data Format

The supported RealNetworks Real Video data format, file extension, and MIME type are as follows:

- Data format: RMFF
- File extension: .rm
- MIME type: video/x-pn-realvideo

## C.4 Supported 3GP Data Format

The supported video 3GP data format, file extension, and MIME type are as follows:

- Data format: 3GP
- File extension: .3gp
- MIME type: video/3gpp

## C.5 Supported Video MPEG Data Formats

The supported video MPEG formats are MPEG1, MPEG2, and MPEG4, as described in the following sections:

- Supported MPEG1 and MPEG2 Data Formats
- Supported MPEG4 Data Formats

### C.5.1 Supported MPEG1 and MPEG2 Data Formats

The supported video MPEG1 and MPEG2 data format, file extension, and MIME type are as follows:

- Data format: MPEG
- File extension: .mpg
- MIME type: video/mpeg

### C.5.2 Supported MPEG4 Data Formats

The supported video MPEG4 data format, file extension, and MIME type are as follows:

- Data format: MP4
- File extension: .mp4
- MIME type: video/mp4

## C.6 Supported Windows Media File Format (ASF) for Video Data

Oracle Multimedia supports the Advanced Systems Format (ASF) file format used by Microsoft Windows Media. These files are denoted by the suffixes .wmv and .asf. The Oracle Multimedia objects ORDVideo and ORDDoc can recognize ASF format data and extract attributes about the video media from the data.

---

**Note:** To access these formats, install the software from the Oracle Multimedia Software section of the Oracle Technology Network Web site at

`http://www.oracle.com/technology/products/multimedia/`

See *Support for the Windows Media File Format (ASF)*.

---

**See Also:**

http://www.microsoft.com/windows/windowsmedia/ for more information about the Advanced Systems Format used by Microsoft Windows Media

The supported Windows Media file extension and MIME type for video data stored in an ORDVideo object are as follows:

- File extension: .wmv
- MIME type: audio/x-ms-wmv

The supported Windows Media file extension and MIME type for video data stored in an ORDDoc object are as follows:

- File extension: .asf
- MIME type: video/x-ms-asf

Table C–3 lists the supported Windows Media video compression format names and format codes for the ASF formats.

***Table C–3 ASF Video Compression Formats and Format Codes***

| Compression Formats | Format Codes |
|---|---|
| Windows Media Video V7 | WMV1 |
| Windows Media Video V8 | WMV2 |
| Windows Media Video 9 | WMV3 |
| Windows Media Screen V7 | MSS1 |
| Windows Media Video 9 Screen | MSS2 |
| Windows Media MPEG-4 Video v3 | MP43 |
| ISO MPEG-4 Video V1 | MP4S |
| Uncompressed YUV422 | YUV2 |

# D

# Oracle Multimedia Image Processing

This appendix describes processing operations for Oracle Multimedia images. It also describes the command options, or operators, used in the Oracle Multimedia process( ) and processCopy( ) methods.

This appendix includes these sections:



## D.1 Common Concepts for Image Processing

This section describes concepts common to all the image operators, and the process( ) and processCopy( ) methods, in the following subsections:

- Source and Destination Images
- process( ) and processCopy( )
- Operator and Value
- Combining Operators
- Types of Image Operators

See Chapter 5 and Chapter 7 for reference information about the process( ) and processCopy( ) methods.

> **Note:** Information about supported image file formats and image compression formats is presented in Appendix B. See Table B–1, Table B–2, and Table B–3, specifically.

### 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.

See process( ) and processCopy( ) for reference information about these methods.

### 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 is to 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 enables 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.1.5 Types of Image Operators

Oracle Multimedia supports three basic types of image operators: image formatting operators, image processing operators, and format-specific operators. Table D–1 lists the operators and identifies them as one of the three types.

***Table D–1 Supported Image Operators and Types***

| Name | Type |
|---|---|
| channelOrder | Format-specific |
| compressionFormat | Image formatting |
| compressionQuality | Image formatting |
| contentFormat | Image formatting |
| contrast | Image processing |
| cut | Image processing |
| fileFormat | Image formatting |
| flip | Image processing |
| gamma | Image processing |
| inputChannels | Format-specific |

*Table D–1 (Cont.) Supported Image Operators and Types*

| Name | Type |
|---|---|
| mirror | Image processing |
| nometadata | Image processing |
| page | Image processing |
| pixelOrder | Format-specific |
| quantize | Image processing |
| rotate | Image processing |
| Scaling Operators: | Image processing |
| fixedScale | |
| maxScale | |
| scale | |
| xScale | |
| yScale | |
| scanlineOrder | Format-specific |
| sharpen | Image processing |
| tiled | Image processing |
| transparencyFillColor | Image processing |

## 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)

The following subsections describe these image formatting operators:

- fileFormat
- contentFormat
- compressionFormat
- compressionQuality

### 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 5–1 in Chapter 5. 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 compression quality is useful, and whether the format-specific operators is useful.

If the fileFormat operator is not used in the process( ) command string, Oracle Multimedia determines the file format of the source image and uses that as the default file format value. If the file format of the source image does not support output, an error occurs. If the source image is a foreign image, the output image is 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 usually uses 32 bits to represent each color.

  If the contentFormat operator is not passed to the process( ) method, then Oracle Multimedia 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.

These 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.3.8 for information about 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 are 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 can be any 24-bit RGB value.

The color portion of the contentFormat syntax controls whether the resulting image is 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 has 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 MEDIANCUT value. See Section D.3.8 for information about the quantize operator.

***Figure D–2 Syntax Diagram for LUT contentFormat***

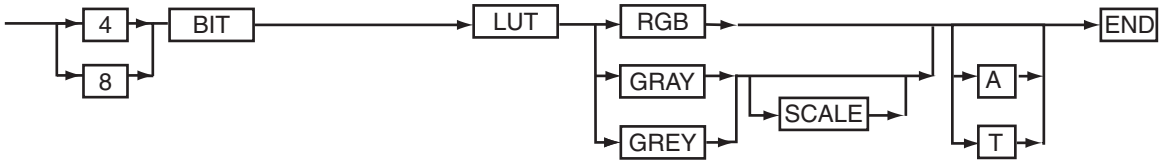



Figure D–3 illustrates the contentFormat syntax that you use to convert an image to grayscale.

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 is always 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***

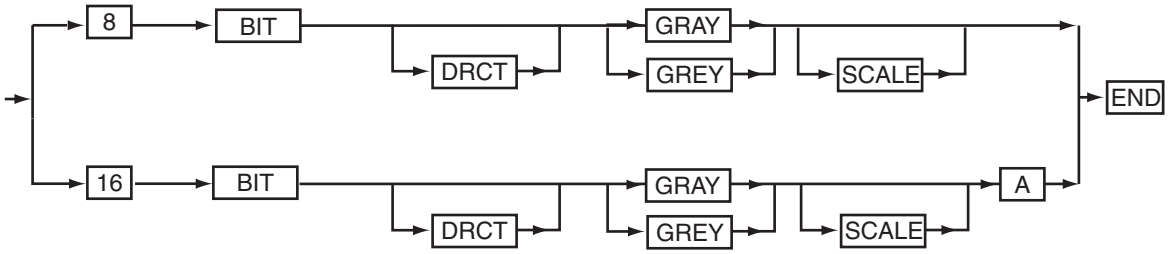


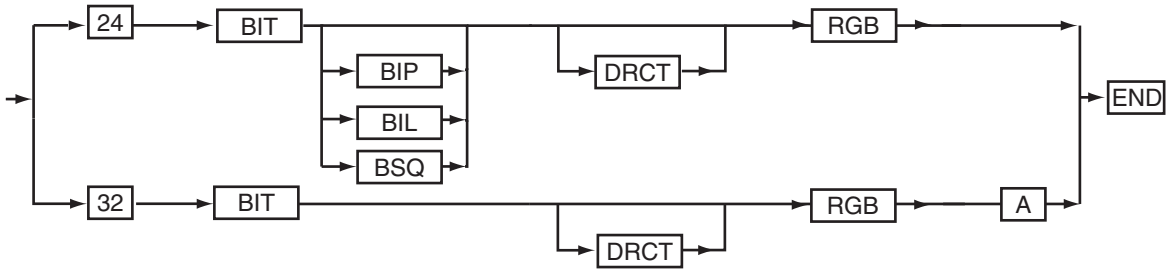

Figure D–4 illustrates the contentFormat syntax that you use to convert an image to direct color.

The bit depth portion of the contentFormat syntax determines the overall type of the direct RGB image: a 24-bit RGB image does 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 is always 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 enables 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***




This 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 5–1 in Chapter 5.

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 that is compressed into a lossless format and then decompressed looks 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 Oracle Multimedia.

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 and the file format of the destination image is either the same as that of the source image or not specified, and if the content format of the destination image is either the same as that of the source image or not specified, then the compression format of the destination image is the same as the source image.

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 is selected depending on the destination image file format. This default compression format 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. File formats that support JPEG compression include JFIF, TIFF, and PICT.

The compressionQuality operator accepts five values, ranging from the most compression (lowest visual quality) to the least compression (highest visual quality): MAXCOMPRATIO, HIGHCOMP, MEDCOMP, LOWCOMP, and MAXINTEGRITY. Using the MAXCOMPRATIO value results in the smallest amount of image data, but may introduce visible aberrations. Using the MAXINTEGRITY value keeps the resulting image more faithful to the original, but requires more space to store. The compressionQuality operator also accepts integer values between 0 (lowest quality) and 100 (highest quality) for JFIF and TIFF file formats only.

The default values for the compressionQuality operator are LOWCOMP for the JFIF and TIFF file formats and MAXINTEGRITY for the PICT file format.

## D.3 Image Processing Operators

The image processing operators supported by Oracle Multimedia directly change the way the image looks on the display. The operators supported by Oracle Multimedia represent only a fraction of all possible image processing operations, and are not intended for users performing intricate image analysis.

The following subsections describe these image processing operators:

- contrast
- cut
- flip
- gamma
- mirror
- nometadata
- page
- quantize
- rotate
- Scaling Operators
- sharpen
- tiled
- transparencyFillColor

### 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 can 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 less than the lower bound are forced to zero intensity, and values greater than the upper bound are forced to full intensity. For 8-bit grayscale and 24-bit RGB images, these bounds can 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.

  **Note:** Enclose all floating-point arguments with double quotation marks (" ") to ensure correct Globalization Support interpretation.

### 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 this 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).

**Note:** Enclose all floating-point arguments with double quotation marks (" ") to ensure correct Globalization Support interpretation.

### 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 nometadata

The nometadata operator processes an image without encoding the image's metadata in the resulting image. This example shows how to generate a JPEG thumbnail image without metadata:

```
image.processCopy('fileformat=jpeg maxscale=100 100 nometadata', dest);
```

### D.3.7 page

The page operator enables page selection from a multipage input image. The value specifies the input page to 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.3.8 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 to perform any resulting quantization (reduction in number of colors).

The value of the quantize operator can be any one of these 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 may 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 causes any input value less than 128 to become black, while the remainder of the image becomes white. A threshold value of 0 causes the entire image to be white, and a value of 256 causes 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 produce better output when converting photographic images to monochrome, but result in fuzziness in synthetic images; using the THRESHOLD quantizer eliminates 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 can be specified. For a sampling rate *n* greater than 1, 1 pixel out of every *n* pixels is examined.

These examples demonstrate how to specify values and arguments for the quantize operator:

```
image.process('contentformat=8bitlutrbg quantize = mediancut 2');
```

```
image.process('contentformat=monochrome quantize = threshold 128');
```

## D.3.9 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.

---

**Note:** Enclose all floating-point arguments with double quotation marks (`" "`) to ensure correct Globalization Support interpretation.

---

## D.3.10 Scaling Operators

Oracle Multimedia supports several operators that change the scale of an image, as described in the following sections.

### D.3.10.1 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 can be larger or smaller (or one larger and one smaller) than the dimensions of the source image.

The scaling method used by this operator is the same as the scaling method used by the scale operator in all cases. This operator cannot be combined with other scaling operators.

### D.3.10.2 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 can 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 has 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 can actually be equal to the values supplied to the operator. The other dimension can 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 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.3.10.3 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 is scaled up (enlarged). If the value is less than 1.0, then the output is 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 Oracle Multimedia. 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.

---

**Note:** Enclose all floating-point arguments with double quotation marks (" ") to ensure correct Globalization Support interpretation.

---

### D.3.10.4 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 can be combined with the yScale operator to scale each axis differently. It cannot be combined with other scaling operators (Scale, fixedScale, maxScale).

---

**Note:** Enclose all floating-point arguments with double quotation marks (" ") to ensure correct Globalization Support interpretation.

---

### D.3.10.5 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 can be combined with the xScale operator to scale each axis differently. It cannot be combined with other scaling operators (scale, fixedScale, maxScale).

**Note:** Enclose all floating-point arguments with double quotation marks (" ") to ensure correct Globalization Support interpretation.

## D.3.11 sharpen

The **sharpen** operator improves the sharpness of an image. For optimum quality, use this operator with image scaling (see Section D.3.10).

The sharpen operator has this syntax:

```
sharpen=kernelType gainFactor
```

`kernelType` is an integer value that specifies the kernel type, which operates as a low-pass filter in image sharpening. The following table shows the valid `kernelType` values and their meanings:

| kernelType Value | Description |
|---|---|
| `0` | Average kernel |
| `1` | Laplacian kernel |
| `2` | Enhanced Laplacian kernel |

The higher the `kernelType` value, the more enhanced the edges of the image become and the sharper the image appears. However, the noise level may increase.

`gainFactor` is a floating-point value that specifies the gain factor used in image sharpening. The higher the `gainFactor` value, the more enhanced the edges of the image become and the sharper the image appears. However, the noise level may increase.

The arguments `kernelType=0` and `gainFactor =2.0` in the following examples produce good results:

```
image.process('maxScale=200 200, sharpen=0 "2.0"');
image.process('sharpen=0 "2.0"');
```

You can experiment with different values for these arguments to determine the combinations that result in the best images for your application.

**Note:** Enclose all floating-point arguments with double quotation marks (" ") to ensure correct Globalization Support interpretation.

## D.3.12 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 depends on the compression format that you select.

## D.3.13 transparencyFillColor

The transparencyFillColor operator fills the transparent regions of PNG image file formats with color. This operator can be used with or without other image processing operators.

This operator accepts either the string value for a valid color or the triple integer RGB values for a valid color. The following table shows the valid values for the transparencyFillColor operator:

| String Value | RGB Value |
|---|---|
| black | 0 0 0 |
| blue | 0 0 255 |
| cyan | 0 255 255 |
| dark_gray | 169 169 169 |
| gray | 128 128 128 |
| green | 0 255 0 |
| light_gray | 211 211 211 |
| magenta | 255 0 255 |
| orange | 255 128 0 |
| pink | 255 192 203 |
| red | 255 0 0 |
| white | 255 255 255 |
| yellow | 255 255 0 |

Both of the following examples fill the transparent regions of the image with the color red:

```
image.process('transparencyFillColor=red');
image.processCopy('transparencyFillColor=255 0 0');
```

## 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), except 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 Oracle Multimedia automatically, because the source format is Raw Pixel, BMPF, or a foreign image.

The following subsections describe these format-specific operators:

- channelOrder
- pixelOrder
- scanlineOrder
- inputChannels

### 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 is written such that this order is not lost.

See Appendix E for more information about the Raw Pixel file format and the ordering of channels in that format.

### 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 appears first in the image data stream. The value Reverse causes the rightmost pixel of the scanline to appear first.

See Appendix E for more information about the Raw Pixel file format and pixel ordering.

### 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 appears 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.

See Appendix E for more information about the Raw Pixel or BMPF file format and scanline ordering.

### 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 can be specified and the selected band is 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.

Every Raw Pixel or foreign image has these input channel assignments written into its header block, but this operator overrides those default assignments.

See Appendix E for more information about the Raw Pixel file format and input channels.

## D.5 CMYK Image Processing

The process( ) and processCopy( ) methods are supported for TIFF and JFIF images with pixel data stored in the CMYK content format. All processing operators including cut, scale, and rotate are supported for CMYK images. During processing, CMYK data is converted to standard RGB, according to these standard formulas.

For 32-bit CMYK images:

```
R = 255*(1-K/255)*(1-C/255)
G = 255*(1-K/255)*(1-M/255)
B = 255*(1-K/255)*(1-Y/255)
```

For 64-bit CMYK images:

```
R = 65535*(1-K/65535)*(1-C/65535)
```

```
G = 65535*(1-K/65535)*(1-M/65535)
B = 65535*(1-K/65535)*(1-Y/65535)
```

See Table B–1 for information about I/O support for images in CMYK content format.

## D.6 Watermarking Operations

Oracle Multimedia supports adding watermarks to source images. Watermarks can contain images or text – for example, corporate logos, copyrights, or digital signatures. With this feature, watermarking can be enforced by the database.

Watermarking is commonly used to prevent misuse of copyrighted or trademarked images. Thus, you might want to add a watermark to your images before posting them on a Web site or other public forum. A watermark that identifies you as the owner of the images can prevent their unauthorized use.

The following subsections describe these watermarking topics in more detail:

- Adding Image Watermarks
- Adding Text Watermarks
- Watermark Properties

See Chapter 5 and Chapter 7 for reference information about the applyWatermark( ) methods.

### D.6.1 Adding Image Watermarks

For an image watermark, you can specify properties such as the location within a source image (either as the relative position to the source image or as X and Y coordinates), the width and height of the frame, and the transparency.

For example, to specify an image watermark to be displayed in the top left corner of an image, use this syntax:

```
prop := ordsys.ord_str_list(
                 'position=topleft');
```

To specify the size of the frame (in pixels) and the transparency of the image watermark, use this syntax:

```
prop := ordsys.ord_str_list(
                 'width=100',
                 'height=80',
                 'transparency=0.2');
```

See Section D.6.3 for a complete list of watermark properties.

See applyWatermark( ) image for reference information about the object method to use for adding image watermarks.

See applyWatermark( ) image for BFILEs and applyWatermark( ) image for BLOBs for reference information about the relational methods.

### D.6.2 Adding Text Watermarks

For a text watermark, you can specify properties such as the location within the source image, the width and height of the frame (which also controls text wrapping and truncation), and the font size, style, color, and transparency of the text.

For example, to specify a copyright text watermark for Oracle Multimedia to be displayed at the X and Y coordinates of `100`, in the font Times New Roman and the style Italic, with `0.6` transparency, use this syntax:

```
added_text := 'Oracle Multimedia © 2009';
 -- specify properties
prop := ordsys.ord_str_list(
                  'font_name=Times New Roman',
                  'font_style=italic',
                  'position_x=100',
                  'position_y=100',
                  'transparency=0.6');
```

See Section D.6.3 for a complete list of watermark properties.

See applyWatermark( ) text for reference information about the object method to use for adding text watermarks.

See applyWatermark( ) text for BFILEs and applyWatermark( ) text for BLOBs for reference information about the relational methods.

## D.6.3 Watermark Properties

Watermark properties are defined as name-value pairs. Table D–2 describes these properties and lists the valid values for each property.

***Table D–2 Watermark Properties and Valid Values***

| Name | Description | Values |
|---|---|---|
| font_name | The name of the font to use for the watermark text | Valid values: The name of any font that is valid for your system. The default is `Arial`. |
| font_style | The style of the font to use for the watermark text | Valid values: `PLAIN`, `BOLD`, and `ITALIC`. The default is `PLAIN`. |
| font_size | The size of the font to use for the watermark text | Valid values: A positive integer. The default is `30`. |
| text_color | The color to use for the watermark text | Valid values: `black`, `blue`, `cyan`, `dark_gray`, `gray`, `green`, `light_gray`, `magenta`, `orange`, `pink`, `red`, `white`, and `yellow`. The default is `blue`. |
| text_color_red, text_color_green, text_color_blue | The red, green, and blue channels of color to use for the watermark text | Valid values: An integer between `0` and `255`. The default is `0`.<br>**Note:** You must specify all three properties at the same time. A valid specification of these three properties overrides a valid specification of the text_color property. |
| position | The relative position of the watermark in the source image | Valid values: `topleft`, `topcenter`, `topright`, `middleleft`, `middlecenter`, `middleright`, `bottomleft`, `bottomcenter`, and `bottomright`. The default is `middlecenter`. |

*Table D–2 (Cont.) Watermark Properties and Valid Values*

| Name | Description | Values |
|---|---|---|
| position_x, position_y | The x and y coordinates for the position of the watermark in the source image | Valid values: Positive integers. The default is `50`.<br><br>**Note:** You must specify both properties at the same time. A valid specification of these two properties overrides a valid specification of the position property. |
| transparency | The transparency of the watermark in the source image | Valid values: A floating-point number between `0` and `1`, where `0` represents full transparency and `1` represents no transparency. The default is `0.5`. |
| width | The width of the watermark (in pixels) in the source image | Valid values: A positive integer. The default is the actual width of the watermark. |
| height | The height of the watermark (in pixels) in the source image | Valid values: A positive integer. The default is the actual height of the watermark.<br><br>**Note:** To wrap long text within a frame, specify the width and the height of the frame. Text that does not fit within the frame is truncated. |

# E

# Image Raw Pixel Format for Use With Oracle Multimedia

This appendix describes the Oracle Raw Pixel image format and is intended for developers and advanced users who want to use the Raw Pixel format to import unsupported image formats into Oracle Multimedia, or as a means to directly access the pixel data in an image.

Much of this appendix is also applicable to foreign images.

This appendix includes these sections:



## E.1 Raw Pixel Introduction

Oracle Multimedia 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 must 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 Oracle Multimedia, but is in a simple, uncompressed format. These users can prepend a Raw Pixel identifier and header onto their data and import it into Oracle Multimedia. For users who need only to read these images (such as for import or conversion), this capability is built into Oracle Multimedia as "Foreign Image Support." Section E.10 describes how this capability is related to the Raw Pixel format.

In addition to supporting image types not built into Oracle Multimedia, 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 can be selected during conversion to another image format, enabling easy visualization within programs that do not otherwise support N-band images. 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 can 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.

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.<br>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.<br>The value of this field can 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. |
| 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. |

*Table E–1 (Cont.) Raw Pixel Image Header Structure*

| Name | Byte(s) | Description |
|---|---|---|
| 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.<br>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.<br>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.<br>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" (hexadecimal 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 enables 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 want 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 can store an image dimension in excess of 4 billion pixels, limitations within Oracle Multimedia require that this field be a value between 1 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 can store an image dimension in excess of 4 billion pixels, limitations within Oracle Multimedia 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 can 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 can 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 can 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. See Section E.5.3 for more information about the meaning of the various interleave options.

This field can 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 cannot 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 number can 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 can 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 is to be selected for display, then the band number must be encoded as the red channel. In this case, the green and blue channels are set to 0.

This field cannot 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 number can 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 can 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 is to be selected for display, then the band number must be encoded as the red channel. In this case, the green and blue channels are set to 0.

This field can 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 number can 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 can 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 is to be selected for display, then the band number must be encoded as the red channel. In this case, the green and blue channels are set to 0.

This field can 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 can 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 is 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 can be presented in the image as either topmost first, or bottommost first. Within a scanline, pixels can be ordered leftmost first, or rightmost first. All these options are affected by interleaving in a relatively straightforward way.

The following subsections provide examples for these topics related to Raw Pixel data:

- Scanline Ordering
- Pixel Ordering
- Band Interleaving
- N-Band Data

## E.5.1 Scanline Ordering

On the screen, an image may look like this:

```
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...
```

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 this:

```
...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...
```

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: RRRRRRRRRRRRRRRRRRRR…
            GGGGGGGGGGGGGGGGGGGG…
            BBBBBBBBBBBBBBBBBBBB…
scanline 2: RRRRRRRRRRRRRRRRRRRR…
            GGGGGGGGGGGGGGGGGGGG…
            BBBBBBBBBBBBBBBBBBBB…
scanline 3: RRRRRRRRRRRRRRRRRRRR…
            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: RRRRRRRRRRRRRRRRRR…  (part of scanline 1)
         RRRRRRRRRRRRRRRRRR…  (part of scanline 2)
         RRRRRRRRRRRRRRRRRR…  (part of scanline 3)
…
plane 2: GGGGGGGGGGGGGGGGGG…  (part of scanline 1)
         GGGGGGGGGGGGGGGGGG…  (part of scanline 2)
         GGGGGGGGGGGGGGGGGG…  (part of scanline 3)
…
plane 3: BBBBBBBBBBBBBBBBBB…  (part of scanline 1)
         BBBBBBBBBBBBBBBBBB…  (part of scanline 2)
```

```
BBBBBBBBBBBBBBBBBB…  (part of scanline 3)
…
```

### E.5.4 N-Band Data

The Raw Pixel format supports up to 255 bands of data in an image. Section E.5.3 describes the relative location of these bands of data in the image, including 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 provided 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<br>*(12121212)*<br>**[12121212]** | 11112222<br>*(11112222)*<br>**[11112222]** | 1111*(1111)***[1111]**<br>2222*(2222)***[2222]** |
| 4 | 1234123412341234<br>*(1234123412341234)*<br>**[1234123412341234]** | 1111222233334444<br>*(1111222233334444)*<br>**[1111222233334444]** | 1111*(1111)***[1111]**<br>2222*(2222)***[2222]**<br>3333*(3333)***[3333]**<br>4444*(4444)***[4444]** |
| 5 | 12345123451234512345<br>*(12345123451234512345)*<br>**[12345123451234512345]** | 11112222333344445555<br>*(11112222333344445555)*<br>**[11112222333344445555]** | 1111*(1111)***[1111]**<br>2222*(2222)***[2222]**<br>3333*(3333)***[3333]**<br>4444*(4444)***[4444]**<br>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:** Various macros for the UNDEFINED values are meant to be descriptive, and should not be used. The exception, "RPIX_ CHANNEL_UNDEFINED," 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 these 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 must understand how to read and write the CCITT formatted data.

## E.10 Foreign Image Support and the Raw Pixel Format

Oracle Multimedia 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 Oracle Multimedia. These rules are summarized in the following sections.

### Header

Foreign images can have any header (or no header), in any format, if its length does not exceed 4,294,967,265 bytes. As has been noted before, all information in this header is ignored.

### Image Width

Foreign images can be up to 32,767 pixels wide.

### Image Height

Foreign images can 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 can 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 can 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 on, 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 permitted, nor can bands have any pixel, scanline, or band-level blocking or padding.

### Number of Bands

Foreign images can 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 cannot have more than 255 bands because the additional bands change the layout of the bitmap data.

### Trailer

Foreign images can have an image trailer following the bitmap data, and this trailer can be of arbitrary length. However, such data is completely ignored by Oracle Multimedia, 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 does not contain this trailer. In the case of the processCopy( ) method, the source image is still intact.

# F

# Oracle Multimedia Metadata XML Schemas

This appendix lists the XML schemas used by the metadata methods of the ORDImage object type. When Oracle Multimedia is installed, these schemas are registered as global XML schemas in Oracle Database with Oracle XML DB.

> **Note:** The schemas in this appendix might not match the code shipped with the Oracle installation. For the final versions of these schemas, use the files provided with the installation.

The latest versions of these schemas are available as files located in the `ord/xml/xsd` directory under *<ORACLE_HOME>*. To examine the schemas, query the dictionary view ALL_XML_SCHEMAS. In addition, read the documentation embedded within each schema file for more information.

This appendix includes these XML schemas:



**See Also:**

- *Oracle XML DB Developer's Guide* for information about registering XML schemas
- `http://www.w3.org/XML/Schema` for more information about XML schemas
- *Oracle Database Reference* for more information about the dictionary view ALL_XML_SCHEMAS

## F.1 XML Schema for EXIF Metadata

This schema is the content model for EXIF metadata retrieved from images. The namespace for this schema is `http://xmlns.oracle.com/ord/meta/exif`.

```
<?xml version="1.0" encoding="UTF-8"?>
<!-- Copyright (c) 2004, 2008, Oracle and/or its affiliates.
All rights reserved. -->
<xsd:schema targetNamespace="http://xmlns.oracle.com/ord/meta/exif"
            xmlns="http://xmlns.oracle.com/ord/meta/exif"
            xmlns:xsd="http://www.w3.org/2001/XMLSchema"
```

```
          elementFormDefault="qualified"
          attributeFormDefault="unqualified"
          version="1.0">

<xsd:annotation>
  <xsd:documentation>

    Introduction

      This is the Oracle Multimedia schema for image metadata stored
      in the EXIF format for digital still cameras. This schema supports
      tags defined up to EXIF version 2.21

      Metadata extracted only from the 0th IFD. For JPEG images, this means
      that the metadata comes from the main image in the file. Metadata is not
      extracted for the thumbnail image (1st IFD).

    Structure

      This schema defines a single global element exifMetadata which
      contains up to four child elements. Each child element contains tags
      from a TIFF IFD directory as defined by the EXIF standard.

            TiffIfd contains tags from the TIFF IFD.
            ExifIfd contains tags from the EXIF IFD.
            GpsIfd  contains tags from the GPS IFD.
            InteroperabilityIfd contains tags from the Interoperability IFD.

      All elements that derive directly from EXIF tags contain a required
      "tag" attribute. The value of this attribute is the Tag ID value as
      defined in the EXIF standard.

    Unsupported tags
      The table below lists tags that are defined by the EXIF standard
      but which the current version of Oracle Multimedia does not
      read from image files. Note that this schema does define data
      models for these tags and future versions of Oracle
      Multimedia may parse these fields from image files. Those tags
      could be represented by documents conforming to this schema.

        These tags are from the TIFF IFD
        tag   301:  TransferFunction
        tag   318:  WhitePoint
        tag   319:  PrimaryChromaticities
        tag   529:  YCbCrCoefficients
        tag   532:  ReferenceWhiteBlack

        tag   273:  StripOffsets
        tag   278:  RowsPerStrip
        tag   279:  StripByteCounts
        tag   513:  JPEGInterChangeFormat
        tag   514:  JPEGInterChangeFormatLength

        These tags are from the EXIF IFD

        tag 34855:  ISOSpeedRatings
        tag 34856:  OECF
        tag 37396:  SubjectArea
        tag 37500:  MakerNote
        tag 41484:  SpatialFrequencyResponse
```

```
        tag 41492:  SubjectLocation
        tag 41730:  CFAPattern
        tag 41995:  DeviceSettingsDescription
        tag 42016:  ImageUniqueID

  </xsd:documentation>
</xsd:annotation>


<!--
  ATTRIBUTE DEFINITIONS
 -->
<xsd:attributeGroup name="exifAttrs">
  <xsd:annotation>
    <xsd:documentation>
       This attribute group defines a single attribute that is required
       for all elements. The tag attribute value is the TIFF tag value
       (in decimal) that is the data source  for the tag.
    </xsd:documentation>
  </xsd:annotation>
  <xsd:attribute name="tag" type="xsd:nonNegativeInteger" use="required"/>
</xsd:attributeGroup>


<!-- BASE TYPE DEFINITIONS
       Base types are formed from the simple XML schema types.
       Sometimes restrictions are added.
       They are extended with the required "tag" attribute
 -->
<xsd:complexType name="positiveIntegerType">
  <xsd:simpleContent>
    <xsd:extension base="xsd:positiveInteger">
      <xsd:attributeGroup ref="exifAttrs"/>
    </xsd:extension>
  </xsd:simpleContent>
</xsd:complexType>

<xsd:complexType name="nonNegativeIntegerType">
  <xsd:simpleContent>
    <xsd:extension base="xsd:nonNegativeInteger">
      <xsd:attributeGroup ref="exifAttrs"/>
    </xsd:extension>
  </xsd:simpleContent>
</xsd:complexType>

<xsd:complexType name="stringType">
  <xsd:simpleContent>
    <xsd:extension base="xsd:string">
      <xsd:attributeGroup ref="exifAttrs"/>
    </xsd:extension>
  </xsd:simpleContent>
</xsd:complexType>

<xsd:complexType name="nonNegativeRealType">
  <xsd:simpleContent>
    <xsd:extension base="nonNegativeReal">
      <xsd:attributeGroup ref="exifAttrs"/>
    </xsd:extension>
  </xsd:simpleContent>
</xsd:complexType>
```

```
<xsd:simpleType name="nonNegativeReal">
  <xsd:restriction base="xsd:float">
    <xsd:minInclusive value="0"/>
  </xsd:restriction>
</xsd:simpleType>

<xsd:complexType name="realType">
  <xsd:simpleContent>
    <xsd:extension base="xsd:float">
      <xsd:attributeGroup ref="exifAttrs"/>
    </xsd:extension>
  </xsd:simpleContent>
</xsd:complexType>

<xsd:complexType name="dateType">
  <xsd:simpleContent>
    <xsd:extension base="xsd:date">
      <xsd:attributeGroup ref="exifAttrs"/>
    </xsd:extension>
  </xsd:simpleContent>
</xsd:complexType>

<xsd:complexType name="dateTimeType">
  <xsd:simpleContent>
    <xsd:extension base="xsd:dateTime">
      <xsd:attributeGroup ref="exifAttrs"/>
    </xsd:extension>
  </xsd:simpleContent>
</xsd:complexType>

<xsd:complexType name="timeType">
  <xsd:simpleContent>
    <xsd:extension base="xsd:time">
      <xsd:attributeGroup ref="exifAttrs"/>
    </xsd:extension>
  </xsd:simpleContent>
</xsd:complexType>


<!-- EXIF TYPE DEFINITIONS
     Generally these types are formed from a simple type that is restricted.
     The simple type is extended with the required 'tag' attribute
     These types form the basis for the document elements
  -->
<xsd:complexType name="colorSpaceType">
  <xsd:simpleContent>
    <xsd:extension base="colorSpace_t">
      <xsd:attributeGroup ref="exifAttrs"/>
    </xsd:extension>
  </xsd:simpleContent>
</xsd:complexType>
<xsd:simpleType name="colorSpace_t">
  <xsd:restriction base="xsd:string">
    <xsd:enumeration value="sRGB"/>
    <xsd:enumeration value="Uncalibrated"/>
  </xsd:restriction>
</xsd:simpleType>

<xsd:complexType name="exposureProgramType">
  <xsd:simpleContent>
```

```
    <xsd:extension base="exposureProgram_t">
      <xsd:attributeGroup ref="exifAttrs"/>
    </xsd:extension>
  </xsd:simpleContent>
</xsd:complexType>
<xsd:simpleType name="exposureProgram_t">
  <xsd:restriction base="xsd:string">
    <xsd:enumeration value="Not defined"/>
    <xsd:enumeration value="Manual"/>
    <xsd:enumeration value="Normal program"/>
    <xsd:enumeration value="Aperture priority"/>
    <xsd:enumeration value="Shutter priority"/>
    <xsd:enumeration value="Creative program"/>
    <xsd:enumeration value="Action program"/>
    <xsd:enumeration value="Portrait mode"/>
    <xsd:enumeration value="Landscape mode"/>
  </xsd:restriction>
</xsd:simpleType>

<xsd:complexType name="meteringModeType">
  <xsd:simpleContent>
    <xsd:extension base="meteringMode_t">
      <xsd:attributeGroup ref="exifAttrs"/>
    </xsd:extension>
  </xsd:simpleContent>
</xsd:complexType>
<xsd:simpleType name="meteringMode_t">
  <xsd:restriction base="xsd:string">
    <xsd:enumeration value="unknown"/>
    <xsd:enumeration value="Average"/>
    <xsd:enumeration value="Center Weighted Average"/>
    <xsd:enumeration value="Spot"/>
    <xsd:enumeration value="MultiSpot"/>
    <xsd:enumeration value="Pattern"/>
    <xsd:enumeration value="Partial"/>
    <xsd:enumeration value="other"/>
  </xsd:restriction>
</xsd:simpleType>

<xsd:complexType name="lightSourceType">
  <xsd:simpleContent>
    <xsd:extension base="lightSource_t">
      <xsd:attributeGroup ref="exifAttrs"/>
    </xsd:extension>
  </xsd:simpleContent>
</xsd:complexType>
<xsd:simpleType name="lightSource_t">
  <xsd:restriction base="xsd:string">
    <xsd:enumeration value="unknown"/>
    <xsd:enumeration value="Daylight"/>
    <xsd:enumeration value="Fluorescent"/>
    <xsd:enumeration value="Tungsten"/>
    <xsd:enumeration value="Flash"/>
    <xsd:enumeration value="Fine weather"/>
    <xsd:enumeration value="Cloudy weather"/>
    <xsd:enumeration value="Shade"/>
    <xsd:enumeration value="Daylight fluorescent"/>
    <xsd:enumeration value="Day white fluorescent"/>
    <xsd:enumeration value="Cool white fluorescent"/>
    <xsd:enumeration value="Standard light A"/>
```

```
      <xsd:enumeration value="Standard light B"/>
      <xsd:enumeration value="Standard light C"/>
      <xsd:enumeration value="D55"/>
      <xsd:enumeration value="D65"/>
      <xsd:enumeration value="D75"/>
      <xsd:enumeration value="D50"/>
      <xsd:enumeration value="ISO studio tungsten"/>
      <xsd:enumeration value="other light source"/>
    </xsd:restriction>
  </xsd:simpleType>

  <xsd:complexType name="flashType">
    <xsd:sequence>
      <xsd:element name="Fired" type="yesNo_t"/>
      <xsd:element name="Return" type="flashReturn_t" minOccurs="0"/>
      <xsd:element name="Mode" type="flashMode_t" minOccurs="0"/>
      <xsd:element name="Function" type="yesNo_t" minOccurs="0"/>
      <xsd:element name="RedEyeReduction" type="yesNo_t" minOccurs="0"/>
    </xsd:sequence>
    <xsd:attributeGroup ref="exifAttrs"/>
  </xsd:complexType>
  <xsd:simpleType name="yesNo_t">
    <xsd:restriction base="xsd:string">
      <xsd:enumeration value="Yes"/>
      <xsd:enumeration value="No"/>
    </xsd:restriction>
  </xsd:simpleType>
  <xsd:simpleType name="flashReturn_t">
    <xsd:restriction base="xsd:string">
      <xsd:enumeration value="No strobe return function"/>
      <xsd:enumeration value="Strobe return not detected"/>
      <xsd:enumeration value="Strobe return detected"/>
    </xsd:restriction>
  </xsd:simpleType>
  <xsd:simpleType name="flashMode_t">
    <xsd:restriction base="xsd:string">
      <xsd:enumeration value="unknown"/>
      <xsd:enumeration value="Compulsory firing"/>
      <xsd:enumeration value="Compulsory suppression"/>
      <xsd:enumeration value="Auto"/>
    </xsd:restriction>
  </xsd:simpleType>

  <xsd:complexType name="resolutionType">
    <xsd:simpleContent>
      <xsd:extension base="resolution_t">
        <xsd:attributeGroup ref="exifAttrs"/>
      </xsd:extension>
    </xsd:simpleContent>
  </xsd:complexType>
  <xsd:simpleType name="resolution_t">
    <xsd:restriction base="xsd:string">
      <xsd:enumeration value="inches"/>
      <xsd:enumeration value="centimeters"/>
    </xsd:restriction>
  </xsd:simpleType>

  <xsd:complexType name="sensingMethodType">
    <xsd:simpleContent>
      <xsd:extension base="sensingMethod_t">
```

```
      <xsd:attributeGroup ref="exifAttrs"/>
    </xsd:extension>
  </xsd:simpleContent>
</xsd:complexType>
<xsd:simpleType name="sensingMethod_t">
  <xsd:restriction base="xsd:string">
    <xsd:enumeration value="Not defined"/>
    <xsd:enumeration value="One-chip color area"/>
    <xsd:enumeration value="Two-chip color area"/>
    <xsd:enumeration value="Three-chip color area"/>
    <xsd:enumeration value="Color-sequential area"/>
    <xsd:enumeration value="Trilinear"/>
    <xsd:enumeration value="Color sequential linear"/>
  </xsd:restriction>
</xsd:simpleType>

<xsd:complexType name="fileSourceType">
  <xsd:simpleContent>
    <xsd:extension base="fileSource_t">
      <xsd:attributeGroup ref="exifAttrs"/>
    </xsd:extension>
  </xsd:simpleContent>
</xsd:complexType>
<xsd:simpleType name="fileSource_t">
  <xsd:restriction base="xsd:string">
    <xsd:enumeration value="others"/>
    <xsd:enumeration value="scanner of transparent type"/>
    <xsd:enumeration value="scanner of reflex type"/>
    <xsd:enumeration value="DSC"/>
  </xsd:restriction>
</xsd:simpleType>

<xsd:complexType name="bitsPerSampleType">
  <xsd:simpleContent>
    <xsd:extension base="bitsPerSample_t">
      <xsd:attributeGroup ref="exifAttrs"/>
    </xsd:extension>
  </xsd:simpleContent>
</xsd:complexType>
<xsd:simpleType name="bitsPerSample_t">
  <xsd:restriction base="xsd:string">
    <xsd:enumeration value="8,8,8"/>
  </xsd:restriction>
</xsd:simpleType>

<xsd:complexType name="compressionType">
  <xsd:simpleContent>
    <xsd:extension base="compression_t">
      <xsd:attributeGroup ref="exifAttrs"/>
    </xsd:extension>
  </xsd:simpleContent>
</xsd:complexType>
<xsd:simpleType name="compression_t">
  <xsd:restriction base="xsd:string">
    <xsd:enumeration value="uncompressed"/>
    <xsd:enumeration value="JPEG"/>
  </xsd:restriction>
</xsd:simpleType>

<xsd:complexType name="photometricInterpretationType">
```

```
  <xsd:simpleContent>
    <xsd:extension base="photometricInterpretation_t">
      <xsd:attributeGroup ref="exifAttrs"/>
    </xsd:extension>
  </xsd:simpleContent>
</xsd:complexType>
<xsd:simpleType name="photometricInterpretation_t">
  <xsd:restriction base="xsd:string">
    <xsd:enumeration value="RGB"/>
    <xsd:enumeration value="YCbCr"/>
  </xsd:restriction>
</xsd:simpleType>

<xsd:complexType name="orientationType">
  <xsd:simpleContent>
    <xsd:extension base="orientation_t">
      <xsd:attributeGroup ref="exifAttrs"/>
    </xsd:extension>
  </xsd:simpleContent>
</xsd:complexType>
<xsd:simpleType name="orientation_t">
  <xsd:restriction base="xsd:string">
    <xsd:enumeration value="top left"/>
    <xsd:enumeration value="top right"/>
    <xsd:enumeration value="bottom right"/>
    <xsd:enumeration value="bottom left"/>
    <xsd:enumeration value="left top"/>
    <xsd:enumeration value="right top"/>
    <xsd:enumeration value="right bottom"/>
    <xsd:enumeration value="left bottom"/>
  </xsd:restriction>
</xsd:simpleType>

<xsd:complexType name="planarConfigurationType">
  <xsd:simpleContent>
    <xsd:extension base="planarConfiguration_t">
      <xsd:attributeGroup ref="exifAttrs"/>
    </xsd:extension>
  </xsd:simpleContent>
</xsd:complexType>
<xsd:simpleType name="planarConfiguration_t">
  <xsd:restriction base="xsd:string">
    <xsd:enumeration value="chunky"/>
    <xsd:enumeration value="planar"/>
  </xsd:restriction>
</xsd:simpleType>

<xsd:complexType name="yCbCrSubSamplingType">
  <xsd:simpleContent>
    <xsd:extension base="yCbCrSubSampling_t">
      <xsd:attributeGroup ref="exifAttrs"/>
    </xsd:extension>
  </xsd:simpleContent>
</xsd:complexType>
<xsd:simpleType name="yCbCrSubSampling_t">
  <xsd:restriction base="xsd:string">
    <xsd:enumeration value="4:2:2"/>
    <xsd:enumeration value="4:2:0"/>
  </xsd:restriction>
</xsd:simpleType>
```

```
<xsd:complexType name="yCbCrPositioningType">
  <xsd:simpleContent>
    <xsd:extension base="yCbCrPositioning_t">
      <xsd:attributeGroup ref="exifAttrs"/>
    </xsd:extension>
  </xsd:simpleContent>
</xsd:complexType>
<xsd:simpleType name="yCbCrPositioning_t">
  <xsd:restriction base="xsd:string">
    <xsd:enumeration value="centered"/>
    <xsd:enumeration value="co-sited"/>
  </xsd:restriction>
</xsd:simpleType>

<xsd:complexType name="stripOffsetsType">
  <xsd:sequence>
    <xsd:element name="StripOffset" maxOccurs="unbounded">
      <xsd:complexType>
        <xsd:sequence>
          <xsd:element name="Index" type="xsd:nonNegativeInteger"/>
          <xsd:element name="Offset" type="xsd:positiveInteger"/>
        </xsd:sequence>
      </xsd:complexType>
    </xsd:element>
  </xsd:sequence>
  <xsd:attributeGroup ref="exifAttrs"/>
</xsd:complexType>

<xsd:complexType name="stripByteCountsType">
  <xsd:sequence>
    <xsd:element name="StripByteCount" maxOccurs="unbounded">
      <xsd:complexType>
        <xsd:sequence>
          <xsd:element name="Index" type="xsd:nonNegativeInteger"/>
          <xsd:element name="Bytes" type="xsd:positiveInteger"/>
        </xsd:sequence>
      </xsd:complexType>
    </xsd:element>
  </xsd:sequence>
  <xsd:attributeGroup ref="exifAttrs"/>
</xsd:complexType>

<xsd:complexType name="whitePointType">
  <xsd:complexContent>
    <xsd:extension base="chromaticity">
      <xsd:attributeGroup ref="exifAttrs"/>
    </xsd:extension>
  </xsd:complexContent>
</xsd:complexType>

<xsd:complexType name="primaryChromaticitiesType">
  <xsd:sequence>
    <xsd:element name="Color_1" type="chromaticity"/>
    <xsd:element name="Color_2" type="chromaticity"/>
    <xsd:element name="Color_3" type="chromaticity"/>
  </xsd:sequence>
  <xsd:attributeGroup ref="exifAttrs"/>
</xsd:complexType>
```

```
<xsd:complexType name="chromaticity">
  <xsd:sequence>
    <xsd:element name="X" type="nonNegativeReal"/>
    <xsd:element name="Y" type="nonNegativeReal"/>
  </xsd:sequence>
</xsd:complexType>

<xsd:complexType name="yCbCrCoefficientsType">
  <xsd:sequence>
    <xsd:element name="Coefficient_1" type="nonNegativeReal"/>
    <xsd:element name="Coefficient_2" type="nonNegativeReal"/>
    <xsd:element name="Coefficient_3" type="nonNegativeReal"/>
  </xsd:sequence>
  <xsd:attributeGroup ref="exifAttrs"/>
</xsd:complexType>

<xsd:complexType name="subjectLocationType">
  <xsd:sequence>
    <xsd:element name="CenterX" type="xsd:nonNegativeInteger"/>
    <xsd:element name="CenterY" type="xsd:nonNegativeInteger"/>
  </xsd:sequence>
  <xsd:attributeGroup ref="exifAttrs"/>
</xsd:complexType>

<xsd:complexType name="subjectAreaType">
  <xsd:complexContent>
    <xsd:extension base="subjectLocationType">
      <xsd:choice>
        <xsd:element name="Diameter" type="xsd:positiveInteger"/>
        <xsd:sequence>
          <xsd:element name="Width" type="xsd:positiveInteger"/>
          <xsd:element name="Height" type="xsd:positiveInteger"/>
        </xsd:sequence>
      </xsd:choice>
    </xsd:extension>
  </xsd:complexContent>
</xsd:complexType>

<xsd:complexType name="customRenderedType">
  <xsd:simpleContent>
    <xsd:extension base="customRendered_t">
      <xsd:attributeGroup ref="exifAttrs"/>
    </xsd:extension>
  </xsd:simpleContent>
</xsd:complexType>
<xsd:simpleType name="customRendered_t">
  <xsd:restriction base="xsd:string">
    <xsd:enumeration value="Normal process"/>
    <xsd:enumeration value="Custom process"/>
  </xsd:restriction>
</xsd:simpleType>

<xsd:complexType name="exposureModeType">
  <xsd:simpleContent>
    <xsd:extension base="exposureMode_t">
      <xsd:attributeGroup ref="exifAttrs"/>
    </xsd:extension>
  </xsd:simpleContent>
</xsd:complexType>
<xsd:simpleType name="exposureMode_t">
```

```
  <xsd:restriction base="xsd:string">
    <xsd:enumeration value="Auto exposure"/>
    <xsd:enumeration value="Manual exposure"/>
    <xsd:enumeration value="Auto bracket"/>
  </xsd:restriction>
</xsd:simpleType>

<xsd:complexType name="whiteBalanceType">
  <xsd:simpleContent>
    <xsd:extension base="whiteBalance_t">
      <xsd:attributeGroup ref="exifAttrs"/>
    </xsd:extension>
  </xsd:simpleContent>
</xsd:complexType>
<xsd:simpleType name="whiteBalance_t">
  <xsd:restriction base="xsd:string">
    <xsd:enumeration value="Auto"/>
    <xsd:enumeration value="Manual"/>
  </xsd:restriction>
</xsd:simpleType>

<xsd:complexType name="sceneCaptureType">
  <xsd:simpleContent>
    <xsd:extension base="sceneCapture_t">
      <xsd:attributeGroup ref="exifAttrs"/>
    </xsd:extension>
  </xsd:simpleContent>
</xsd:complexType>
<xsd:simpleType name="sceneCapture_t">
  <xsd:restriction base="xsd:string">
    <xsd:enumeration value="Standard"/>
    <xsd:enumeration value="Landscape"/>
    <xsd:enumeration value="Portrait"/>
    <xsd:enumeration value="Night scene"/>
    <xsd:enumeration value=""/>
  </xsd:restriction>
</xsd:simpleType>

<xsd:complexType name="gainControlType">
  <xsd:simpleContent>
    <xsd:extension base="gainControl_t">
      <xsd:attributeGroup ref="exifAttrs"/>
    </xsd:extension>
  </xsd:simpleContent>
</xsd:complexType>
<xsd:simpleType name="gainControl_t">
  <xsd:restriction base="xsd:string">
    <xsd:enumeration value="None"/>
    <xsd:enumeration value="Low gain up"/>
    <xsd:enumeration value="High gain up"/>
    <xsd:enumeration value="Low gain down"/>
    <xsd:enumeration value="High gain down"/>
  </xsd:restriction>
</xsd:simpleType>

<xsd:complexType name="contrastType">
  <xsd:simpleContent>
    <xsd:extension base="contrast_t">
      <xsd:attributeGroup ref="exifAttrs"/>
    </xsd:extension>
```

```
  </xsd:simpleContent>
</xsd:complexType>
<xsd:simpleType name="contrast_t">
  <xsd:restriction base="xsd:string">
    <xsd:enumeration value="Normal"/>
    <xsd:enumeration value="Soft"/>
    <xsd:enumeration value="Hard"/>
  </xsd:restriction>
</xsd:simpleType>

<xsd:complexType name="saturationType">
  <xsd:simpleContent>
    <xsd:extension base="saturation_t">
      <xsd:attributeGroup ref="exifAttrs"/>
    </xsd:extension>
  </xsd:simpleContent>
</xsd:complexType>
<xsd:simpleType name="saturation_t">
  <xsd:restriction base="xsd:string">
    <xsd:enumeration value="Normal"/>
    <xsd:enumeration value="Low saturation"/>
    <xsd:enumeration value="High saturation"/>
  </xsd:restriction>
</xsd:simpleType>

<xsd:complexType name="sharpnessType">
  <xsd:simpleContent>
    <xsd:extension base="contrast_t">
      <xsd:attributeGroup ref="exifAttrs"/>
    </xsd:extension>
  </xsd:simpleContent>
</xsd:complexType>

<xsd:complexType name="subjectDistanceRangeType">
  <xsd:simpleContent>
    <xsd:extension base="subjectDistanceRange_t">
      <xsd:attributeGroup ref="exifAttrs"/>
    </xsd:extension>
  </xsd:simpleContent>
</xsd:complexType>
<xsd:simpleType name="subjectDistanceRange_t">
  <xsd:restriction base="xsd:string">
    <xsd:enumeration value="unknown"/>
    <xsd:enumeration value="Macro"/>
    <xsd:enumeration value="Close view"/>
    <xsd:enumeration value="Distant view"/>
  </xsd:restriction>
</xsd:simpleType>

<xsd:complexType name="uuidType">
  <xsd:simpleContent>
    <xsd:extension base="uuid_t">
      <xsd:attributeGroup ref="exifAttrs"/>
    </xsd:extension>
  </xsd:simpleContent>
</xsd:complexType>
<xsd:simpleType name="uuid_t">
  <xsd:restriction base="xsd:hexBinary">
    <xsd:pattern value=".{8}-.{4}-.{4}-.{4}-.{12}"/>
  </xsd:restriction>
```

```
</xsd:simpleType>

<!--
     TYPES FOR THE GPS IFD
 -->
<xsd:complexType name="gpsLatitudeRefType">
  <xsd:simpleContent>
    <xsd:extension base="gpsLatitudeRef_t">
      <xsd:attributeGroup ref="exifAttrs"/>
    </xsd:extension>
  </xsd:simpleContent>
</xsd:complexType>
<xsd:simpleType name="gpsLatitudeRef_t">
  <xsd:restriction base="xsd:string">
    <xsd:enumeration value="North latitude"/>
    <xsd:enumeration value="South latitude"/>
  </xsd:restriction>
</xsd:simpleType>

<xsd:complexType name="gpsAltitudeRefType">
  <xsd:simpleContent>
    <xsd:extension base="gpsAltitudeRef_t">
      <xsd:attributeGroup ref="exifAttrs"/>
    </xsd:extension>
  </xsd:simpleContent>
</xsd:complexType>
<xsd:simpleType name="gpsAltitudeRef_t">
  <xsd:restriction base="xsd:string">
    <xsd:enumeration value="Sea level"/>
    <xsd:enumeration value="Sea level reference"/>
  </xsd:restriction>
</xsd:simpleType>

<xsd:complexType name="gpsLatitudeType">
  <xsd:simpleContent>
    <xsd:extension base="gpsLatitude_t">
      <xsd:attributeGroup ref="exifAttrs"/>
    </xsd:extension>
  </xsd:simpleContent>
</xsd:complexType>
<xsd:simpleType name="gpsLatitude_t">
  <xsd:restriction base="xsd:float">
    <xsd:minInclusive value="0.0"/>
    <xsd:maxInclusive value="90.0"/>
  </xsd:restriction>
</xsd:simpleType>

<xsd:complexType name="gpsLongitudeRefType">
  <xsd:simpleContent>
    <xsd:extension base="gpsLongitudeRef_t">
      <xsd:attributeGroup ref="exifAttrs"/>
    </xsd:extension>
  </xsd:simpleContent>
</xsd:complexType>
<xsd:simpleType name="gpsLongitudeRef_t">
  <xsd:restriction base="xsd:string">
    <xsd:enumeration value="East longitude"/>
    <xsd:enumeration value="West longitude"/>
  </xsd:restriction>
</xsd:simpleType>
```

```
<xsd:complexType name="gpsLongitudeType">
  <xsd:simpleContent>
    <xsd:extension base="gpsLongitude_t">
      <xsd:attributeGroup ref="exifAttrs"/>
    </xsd:extension>
  </xsd:simpleContent>
</xsd:complexType>
<xsd:simpleType name="gpsLongitude_t">
  <xsd:restriction base="xsd:float">
    <xsd:minInclusive value="0.0"/>
    <xsd:maxInclusive value="180.0"/>
  </xsd:restriction>
</xsd:simpleType>

<xsd:complexType name="gpsBearingType">
  <xsd:simpleContent>
    <xsd:extension base="gpsBearing_t">
      <xsd:attributeGroup ref="exifAttrs"/>
    </xsd:extension>
  </xsd:simpleContent>
</xsd:complexType>
<xsd:simpleType name="gpsBearing_t">
  <xsd:restriction base="xsd:float">
    <xsd:minInclusive value="0.0"/>
    <xsd:maxExclusive value="360.0"/>
  </xsd:restriction>
</xsd:simpleType>

<xsd:complexType name="gpsStatusType">
  <xsd:simpleContent>
    <xsd:extension base="gpsStatus_t">
      <xsd:attributeGroup ref="exifAttrs"/>
    </xsd:extension>
  </xsd:simpleContent>
</xsd:complexType>
<xsd:simpleType name="gpsStatus_t">
  <xsd:restriction base="xsd:string">
    <xsd:enumeration value="Measurement in progress"/>
    <xsd:enumeration value="Measurement interoperability"/>
  </xsd:restriction>
</xsd:simpleType>

<xsd:complexType name="gpsMeasureModeType">
  <xsd:simpleContent>
    <xsd:extension base="gpsMeasureMode_t">
      <xsd:attributeGroup ref="exifAttrs"/>
    </xsd:extension>
  </xsd:simpleContent>
</xsd:complexType>
<xsd:simpleType name="gpsMeasureMode_t">
  <xsd:restriction base="xsd:string">
    <xsd:enumeration value="2-dimensional measurement"/>
    <xsd:enumeration value="3-dimensional measurement"/>
  </xsd:restriction>
</xsd:simpleType>

<xsd:complexType name="gpsSpeedRefType">
  <xsd:simpleContent>
    <xsd:extension base="gpsSpeedRef_t">
```

```
      <xsd:attributeGroup ref="exifAttrs"/>
    </xsd:extension>
  </xsd:simpleContent>
</xsd:complexType>
<xsd:simpleType name="gpsSpeedRef_t">
  <xsd:restriction base="xsd:string">
    <xsd:enumeration value="Kilometers per hour"/>
    <xsd:enumeration value="Miles per hour"/>
    <xsd:enumeration value="Knots"/>
  </xsd:restriction>
</xsd:simpleType>

<xsd:complexType name="gpsDirectionType">
  <xsd:simpleContent>
    <xsd:extension base="gpsDirection_t">
      <xsd:attributeGroup ref="exifAttrs"/>
    </xsd:extension>
  </xsd:simpleContent>
</xsd:complexType>
<xsd:simpleType name="gpsDirection_t">
  <xsd:restriction base="xsd:string">
    <xsd:enumeration value="True direction"/>
    <xsd:enumeration value="Magnetic direction"/>
  </xsd:restriction>
</xsd:simpleType>

<xsd:complexType name="gpsDistanceRefType">
  <xsd:simpleContent>
    <xsd:extension base="gpsDistanceRef_t">
      <xsd:attributeGroup ref="exifAttrs"/>
    </xsd:extension>
  </xsd:simpleContent>
</xsd:complexType>
<xsd:simpleType name="gpsDistanceRef_t">
  <xsd:restriction base="xsd:string">
    <xsd:enumeration value="Kilometers"/>
    <xsd:enumeration value="Miles"/>
    <xsd:enumeration value="Knots"/>
  </xsd:restriction>
</xsd:simpleType>

<xsd:complexType name="gpsDifferentialType">
  <xsd:simpleContent>
    <xsd:extension base="gpsDifferential_t">
      <xsd:attributeGroup ref="exifAttrs"/>
    </xsd:extension>
  </xsd:simpleContent>
</xsd:complexType>
<xsd:simpleType name="gpsDifferential_t">
  <xsd:restriction base="xsd:string">
    <xsd:enumeration value="Measurement without differential correction"/>
    <xsd:enumeration value="Differential correction applied"/>
  </xsd:restriction>
</xsd:simpleType>

<!--
     TYPES FOR THE INTEROPERABILTY IFD
   -->
<xsd:complexType name="interoperabilityType">
  <xsd:simpleContent>
```

```
    <xsd:extension base="interoperability_t">
      <xsd:attributeGroup ref="exifAttrs"/>
    </xsd:extension>
  </xsd:simpleContent>
</xsd:complexType>
<xsd:simpleType name="interoperability_t">
  <xsd:restriction base="xsd:string">
    <xsd:enumeration value="R98"/>
    <xsd:enumeration value="THM"/>
    <xsd:enumeration value="R03"/>
  </xsd:restriction>
</xsd:simpleType>


<!-- GENERIC TYPE

      A generic type to hold any type of tag data.
      Defines a name, value, datatype triplet
      Datatype values refer to types as defined by XML Schema.

      singleFieldType is for EXIF tag that define a single
      value, the common case.

      repeatedFieldTyp is for the uncommon case where many
      data items are defined in an EXIF tag.

  -->
<xsd:complexType name="singleFieldType">
  <xsd:complexContent>
    <xsd:extension base="singleField_t">
      <xsd:attributeGroup ref="exifAttrs"/>
    </xsd:extension>
  </xsd:complexContent>
</xsd:complexType>

<xsd:complexType name="repeatedFieldType">
  <xsd:sequence>
    <xsd:element name="Field" type="singleField_t" minOccurs="1"
        maxOccurs="unbounded"/>
  </xsd:sequence>
  <xsd:attributeGroup ref="exifAttrs"/>
</xsd:complexType>

<xsd:complexType name="singleField_t">
  <xsd:sequence>
    <xsd:element name="Name" type="xsd:string"/>
    <xsd:element name="Value" type="xsd:string"/>
    <xsd:element name="Datatype">
      <xsd:simpleType>
        <xsd:annotation>
          <xsd:documentation>The enumerated datatype values refer to types
              defined by XML Schema
          </xsd:documentation>
        </xsd:annotation>
        <xsd:restriction base="xsd:string">
          <xsd:enumeration value="string"/>
          <xsd:enumeration value="integer"/>
          <xsd:enumeration value="float"/>
          <xsd:enumeration value="date"/>
          <xsd:enumeration value="time"/>
```

```
          <xsd:enumeration value="dateTime"/>
          <xsd:enumeration value="hexBinary"/>
        </xsd:restriction>
      </xsd:simpleType>
    </xsd:element>
  </xsd:sequence>
</xsd:complexType>

<!-- END TYPE DEFINITIONS -->

<!--
     THE GLOBAL ELEMENT
  -->
<xsd:element name="exifMetadata">
  <xsd:complexType>
    <xsd:sequence>
      <xsd:element name="TiffIfd" minOccurs="0">
        <xsd:annotation>
          <xsd:documentation>Tags from the TIFF IFD</xsd:documentation>
        </xsd:annotation>
        <xsd:complexType>
          <xsd:all>

            <!-- Tags relating to image data structure -->
            <xsd:element name="ImageWidth" type="positiveIntegerType"
                minOccurs="0"/>
            <xsd:element name="ImageLength" type="positiveIntegerType"
                minOccurs="0"/>
            <xsd:element name="BitsPerSample" type="bitsPerSampleType"
                minOccurs="0"/>
            <xsd:element name="Compression" type="compressionType"
                minOccurs="0"/>
            <xsd:element name="PhotometricInterpretation"
                type="photometricInterpretationType" minOccurs="0"/>
            <xsd:element name="Orientation" type="orientationType"
                minOccurs="0"/>
            <xsd:element name="SamplesPerPixel" type="positiveIntegerType"
                minOccurs="0"/>
            <xsd:element name="PlanarConfiguration"
                type="planarConfigurationType" minOccurs="0"/>
            <xsd:element name="YCbCrSubSampling" type="yCbCrSubSamplingType"
                minOccurs="0"/>
            <xsd:element name="YCbCrPositioning" type="yCbCrPositioningType"
                minOccurs="0"/>
            <xsd:element name="XResolution" type="nonNegativeRealType"
                minOccurs="0">
              <xsd:annotation>
                <xsd:documentation>Unit is pixels per
                    ResolutionUnit</xsd:documentation>
              </xsd:annotation>
            </xsd:element>
            <xsd:element name="YResolution" type="nonNegativeRealType"
                minOccurs="0">
              <xsd:annotation>
                <xsd:documentation>Unit is pixels per
                    ResolutionUnit</xsd:documentation>
              </xsd:annotation>
            </xsd:element>
            <xsd:element name="ResolutionUnit" type="resolutionType"
                minOccurs="0"/>
```

```
      <!--Tags relating to recording offset -->
      <xsd:element name="StripOffsets" type="stripOffsetsType"
          minOccurs="0"/>
      <xsd:element name="RowsPerStrip" type="positiveIntegerType"
          minOccurs="0"/>
      <xsd:element name="StripByteCounts" type="stripByteCountsType"
          minOccurs="0"/>
      <xsd:element name="JPEGInterChangeFormat" type="positiveIntegerType"
          minOccurs="0"/>
      <xsd:element name="JPEGInterChangeFormatLength"
          type="positiveIntegerType" minOccurs="0"/>

      <!-- Tags relating to image data characteristics -->
      <xsd:element name="TransferFunction" type="xsd:anyType"
          minOccurs="0"/>
      <xsd:element name="WhitePoint" type="whitePointType" minOccurs="0"/>
      <xsd:element name="PrimaryChromaticities"
          type="primaryChromaticitiesType" minOccurs="0"/>
      <xsd:element name="YCbCrCoefficients" type="yCbCrCoefficientsType"
          minOccurs="0"/>
      <xsd:element name="ReferenceBlackWhite"
          type="primaryChromaticitiesType" minOccurs="0"/>

      <!--Other tags -->
      <xsd:element name="DateTime" type="dateTimeType" minOccurs="0"/>
      <xsd:element name="ImageDescription" type="stringType"
          minOccurs="0"/>
      <xsd:element name="Make" type="stringType" minOccurs="0"/>
      <xsd:element name="Model" type="stringType" minOccurs="0"/>
      <xsd:element name="Software" type="stringType" minOccurs="0"/>
      <xsd:element name="Artist" type="stringType" minOccurs="0"/>
      <xsd:element name="Copyright" type="stringType" minOccurs="0"/>

      <!-- Placeholder tags for future tags that may be defined -->
      <xsd:element name="TiffField1" type="singleFieldType"
          minOccurs="0"/>
      <xsd:element name="TiffField2" type="singleFieldType"
          minOccurs="0"/>
      <xsd:element name="TiffField3" type="repeatedFieldType"
          minOccurs="0"/>

    </xsd:all>
  </xsd:complexType>
</xsd:element>

<xsd:element name="ExifIfd">
  <xsd:annotation>
    <xsd:documentation>Tags from the EXIF IFD</xsd:documentation>
  </xsd:annotation>
  <xsd:complexType>
    <xsd:all>

      <!-- Tags releating to version -->
      <xsd:element name="ExifVersion" type="stringType" minOccurs="0"/>
      <xsd:element name="FlashpixVersion" type="stringType"
          minOccurs="0"/>

      <!-- Tags relating to image data characteristics -->
      <xsd:element name="ColorSpace" type="colorSpaceType" minOccurs="0"/>
```

```
<!-- Tags relating to image configuration -->
<xsd:element name="ComponentsConfiguration" type="stringType"
    minOccurs="0"/>
<xsd:element name="CompressedBitsPerPixel"
    type="nonNegativeRealType" minOccurs="0"/>
<xsd:element name="PixelXDimension" type="nonNegativeIntegerType"
    minOccurs="0"/>
<xsd:element name="PixelYDimension" type="nonNegativeIntegerType"
    minOccurs="0"/>

<!-- Tags relating to user information -->
<xsd:element name="MakerNote" type="repeatedFieldType"
    minOccurs="0"/>
<xsd:element name="UserComment" type="stringType" minOccurs="0"/>

<!-- Tag relating to related file information -->
<xsd:element name="RelatedSoundFile" type="stringType"
    minOccurs="0"/>

<!-- Tags relating to date and time -->
<xsd:element name="DateTimeOriginal" type="dateTimeType"
    minOccurs="0"/>
<xsd:element name="DateTimeDigitized" type="dateTimeType"
    minOccurs="0"/>
<xsd:element name="SubSecTime" type="nonNegativeIntegerType"
    minOccurs="0"/>
<xsd:element name="SubSecTimeOriginal" type="nonNegativeIntegerType"
    minOccurs="0"/>
<xsd:element name="SubSecTimeDigitized"
    type="nonNegativeIntegerType" minOccurs="0"/>

<!-- Tags relating to picture taking conditions -->
<xsd:element name="ExposureTime" type="nonNegativeRealType"
    minOccurs="0">
  <xsd:annotation>
    <xsd:documentation>Units is seconds</xsd:documentation>
  </xsd:annotation>
</xsd:element>
<xsd:element name="FNumber" type="nonNegativeRealType"
    minOccurs="0"/>
<xsd:element name="ExposureProgram" type="exposureProgramType"
    minOccurs="0"/>
<xsd:element name="SpectralSensitivity" type="stringType"
    minOccurs="0"/>
<xsd:element name="ISOSpeedRatings" type="nonNegativeIntegerType"
    minOccurs="0"/>
<xsd:element name="OECF" type="repeatedFieldType" minOccurs="0"/>
<xsd:element name="ShutterSpeedValue" type="realType" minOccurs="0">
  <xsd:annotation>
    <xsd:documentation>The unit is the APEX
        value</xsd:documentation>
  </xsd:annotation>
</xsd:element>
<xsd:element name="ApertureValue" type="nonNegativeRealType"
    minOccurs="0">
  <xsd:annotation>
    <xsd:documentation>The  unit is the APEX
        value</xsd:documentation>
  </xsd:annotation>
```

```
</xsd:element>
<xsd:element name="BrightnessValue" type="realType" minOccurs="0">
  <xsd:annotation>
    <xsd:documentation>The unit is the APEX
        value</xsd:documentation>
  </xsd:annotation>
</xsd:element>
<xsd:element name="ExposureBiasValue" type="realType" minOccurs="0">
  <xsd:annotation>
    <xsd:documentation>The unit is the APEX
        value</xsd:documentation>
  </xsd:annotation>
</xsd:element>
<xsd:element name="MaxApertureValue" type="nonNegativeRealType"
    minOccurs="0">
  <xsd:annotation>
    <xsd:documentation>The unit is the APEX
        value</xsd:documentation>
  </xsd:annotation>
</xsd:element>
<xsd:element name="SubjectDistance" type="stringType" minOccurs="0">
  <xsd:annotation>
    <xsd:documentation>The unit is meters</xsd:documentation>
  </xsd:annotation>
</xsd:element>
<xsd:element name="MeteringMode" type="meteringModeType"
    minOccurs="0"/>
<xsd:element name="LightSource" type="lightSourceType"
    minOccurs="0"/>
<xsd:element name="Flash" type="flashType" minOccurs="0"/>
<xsd:element name="FocalLength" type="nonNegativeRealType"
    minOccurs="0">
  <xsd:annotation>
    <xsd:documentation>The unit is millimeters.</xsd:documentation>
  </xsd:annotation>
</xsd:element>
<xsd:element name="SubjectArea" type="subjectAreaType"
    minOccurs="0"/>
<xsd:element name="FlashEnergy" type="nonNegativeRealType"
    minOccurs="0">
  <xsd:annotation>
    <xsd:documentation>The unit is Beam Candle Power
        Seconds</xsd:documentation>
  </xsd:annotation>
</xsd:element>
<xsd:element name="SpatialFrequencyResponse"
    type="repeatedFieldType" minOccurs="0">
  <xsd:annotation>
    <xsd:documentation>Not implemented</xsd:documentation>
  </xsd:annotation>
</xsd:element>
<xsd:element name="FocalPlaneXResolution" type="nonNegativeRealType"
    minOccurs="0">
  <xsd:annotation>
    <xsd:documentation>The unit is pixels</xsd:documentation>
  </xsd:annotation>
</xsd:element>
<xsd:element name="FocalPlaneYResolution" type="nonNegativeRealType"
    minOccurs="0">
  <xsd:annotation>
```

```
            <xsd:documentation>The unit is pixels</xsd:documentation>
          </xsd:annotation>
        </xsd:element>
        <xsd:element name="FocalPlaneResolutionUnit" type="resolutionType"
            minOccurs="0"/>
        <xsd:element name="SubjectLocation" type="subjectLocationType"
            minOccurs="0"/>
        <xsd:element name="ExposureIndex" type="nonNegativeRealType"
            minOccurs="0"/>
        <xsd:element name="SensingMethod" type="sensingMethodType"
            minOccurs="0"/>
        <xsd:element name="FileSource" type="fileSourceType" minOccurs="0"/>
        <xsd:element name="SceneType" type="stringType" minOccurs="0"/>
        <xsd:element name="CFAPattern" type="repeatedFieldType"
            minOccurs="0"/>
        <xsd:element name="CustomRendered" type="customRenderedType"
            minOccurs="0"/>
        <xsd:element name="ExposureMode" type="exposureModeType"
            minOccurs="0"/>
        <xsd:element name="WhiteBalance" type="whiteBalanceType"
            minOccurs="0"/>
        <xsd:element name="DigitalZoomRatio" type="nonNegativeRealType"
            minOccurs="0"/>
        <xsd:element name="FocalLengthIn35mmFilm" type="positiveIntegerType"
            minOccurs="0"/>
        <xsd:element name="SceneCaptureType" type="sceneCaptureType"
            minOccurs="0"/>
        <xsd:element name="GainControl" type="gainControlType"
            minOccurs="0"/>
        <xsd:element name="Contrast" type="contrastType" minOccurs="0"/>
        <xsd:element name="Saturation" type="saturationType" minOccurs="0"/>
        <xsd:element name="Sharpness" type="sharpnessType" minOccurs="0"/>
        <xsd:element name="DeviceSettingDescription"
            type="repeatedFieldType" minOccurs="0"/>
        <xsd:element name="SubjectDistanceRange"
            type="subjectDistanceRangeType" minOccurs="0"/>
        <xsd:element name="ImageUniqueID" type="uuidType" minOccurs="0"/>
        <xsd:element name="Gamma" type="nonNegativeRealType" minOccurs="0"/>

        <!-- Placeholder tags for future tags that may be defined -->
        <xsd:element name="ExifField1" type="singleFieldType"
            minOccurs="0"/>
        <xsd:element name="ExifField2" type="singleFieldType"
            minOccurs="0"/>
        <xsd:element name="ExifField3" type="repeatedFieldType"
            minOccurs="0"/>

      </xsd:all>
      <xsd:attributeGroup ref="exifAttrs"/>
    </xsd:complexType>
  </xsd:element>

  <xsd:element name="GpsIfd" minOccurs="0">
    <xsd:annotation>
      <xsd:documentation>Tags from the GPS IFD</xsd:documentation>
    </xsd:annotation>
    <xsd:complexType>
      <xsd:all>
        <xsd:element name="GPSVersionID" type="stringType" minOccurs="0"/>
        <xsd:element name="GPSLatitudeRef" type="gpsLatitudeRefType"
```

```
    minOccurs="0"/>
<xsd:element name="GPSLatitude" type="gpsLatitudeType"
    minOccurs="0">
  <xsd:annotation>
    <xsd:documentation>Unit is decimal degrees</xsd:documentation>
  </xsd:annotation>
</xsd:element>
<xsd:element name="GPSLongitudeRef" type="gpsLongitudeRefType"
    minOccurs="0"/>
<xsd:element name="GPSLongitude" type="gpsLongitudeType"
    minOccurs="0">
  <xsd:annotation>
    <xsd:documentation>Unit is decimal degrees</xsd:documentation>
  </xsd:annotation>
</xsd:element>
<xsd:element name="GPSAltitudeRef" type="gpsAltitudeRefType"
    minOccurs="0"/>
<xsd:element name="GPSAltitude" type="nonNegativeRealType"
    minOccurs="0">
  <xsd:annotation>
    <xsd:documentation>Unit is meters</xsd:documentation>
  </xsd:annotation>
</xsd:element>
<xsd:element name="GPSTimeStamp" type="timeType" minOccurs="0"/>
<xsd:element name="GPSSatellites" type="stringType" minOccurs="0"/>
<xsd:element name="GPSStatus" type="gpsStatusType" minOccurs="0"/>
<xsd:element name="GPSMeasureMode" type="gpsMeasureModeType"
    minOccurs="0"/>
<xsd:element name="GPSDOP" type="nonNegativeRealType"
    minOccurs="0"/>
<xsd:element name="GPSSpeedRef" type="gpsSpeedRefType"
    minOccurs="0"/>
<xsd:element name="GPSSpeed" type="nonNegativeRealType"
    minOccurs="0"/>
<xsd:element name="GPSTrackRef" type="gpsDirectionType"
    minOccurs="0"/>
<xsd:element name="GPSTrack" type="gpsBearingType" minOccurs="0"/>
<xsd:element name="GPSImgDirectionRef" type="gpsDirectionType"
    minOccurs="0"/>
<xsd:element name="GPSImgDirection" type="gpsBearingType"
    minOccurs="0"/>
<xsd:element name="GPSMapDatum" type="stringType" minOccurs="0"/>
<xsd:element name="GPSDestLatitudeRef" type="gpsLatitudeRefType"
    minOccurs="0"/>
<xsd:element name="GPSDestLatitude" type="gpsLatitudeType"
    minOccurs="0"/>
<xsd:element name="GPSDestLongitudeRef" type="gpsLongitudeRefType"
    minOccurs="0"/>
<xsd:element name="GPSDestLongitude" type="gpsLongitudeType"
    minOccurs="0"/>
<xsd:element name="GPSDestBearingRef" type="gpsDirectionType"
    minOccurs="0"/>
<xsd:element name="GPSDestBearing" type="gpsBearingType"
    minOccurs="0"/>
<xsd:element name="GPSDestDistanceRef" type="gpsDistanceRefType"
    minOccurs="0"/>
<xsd:element name="GPSDestDistance" type="nonNegativeRealType"
    minOccurs="0"/>
<xsd:element name="GPSProcessingMethod" type="stringType"
    minOccurs="0"/>
```

```
            <xsd:element name="GPSAreaInformation" type="stringType"
                minOccurs="0"/>
            <xsd:element name="GPSDateStamp" type="dateType" minOccurs="0"/>
            <xsd:element name="GPSDifferential" type="gpsDifferentialType"
                minOccurs="0"/>

            <!-- Placeholder tags for future tags that may be defined -->
            <xsd:element name="GPSField1" type="singleFieldType" minOccurs="0"/>
            <xsd:element name="GPSField2" type="singleFieldType" minOccurs="0"/>
            <xsd:element name="GPSField3" type="repeatedFieldType"
                minOccurs="0"/>
          </xsd:all>
          <xsd:attributeGroup ref="exifAttrs"/>
        </xsd:complexType>
      </xsd:element>

      <xsd:element name="InteroperabilityIfd" minOccurs="0">
        <xsd:annotation>
          <xsd:documentation>Tags from the Interoperability
              IFD</xsd:documentation>
        </xsd:annotation>
        <xsd:complexType>
          <xsd:all>
            <xsd:element name="InteroperabilityIndex"
                type="interoperabilityType" minOccurs="0"/>
          </xsd:all>
          <xsd:attributeGroup ref="exifAttrs"/>
        </xsd:complexType>
      </xsd:element>

    </xsd:sequence>
  </xsd:complexType>
 </xsd:element>
</xsd:schema>
```

## F.2 XML Schema for IPTC-IIM Metadata

This schema is the content model for IPTC-IIM metadata retrieved from images. The namespace for this schema is `http://xmlns.oracle.com/ord/meta/iptc`.

```
<?xml version="1.0" encoding="UTF-8"?>
<!-- Copyright (c) 2004, 2008, Oracle. All rights reserved.  -->
<xsd:schema  xmlns="http://xmlns.oracle.com/ord/meta/iptc"
     targetNamespace="http://xmlns.oracle.com/ord/meta/iptc"
     xmlns:xsd="http://www.w3.org/2001/XMLSchema"
     elementFormDefault="qualified" attributeFormDefault="unqualified">

  <xsd:annotation>
    <xsd:documentation>

      Introduction
        This is the Oracle Multimedia schema for metadata stored in Application
        record sets of the IPTC-NAA Information Interchange Model Version 4.
        This metadata is often referred to as 'IPTC' tags.

        For the JPEG file format, IPTC metadata is parsed from the APP13 marker.
        For the TIFF file format, IPTC metadata is parsed from tag 33723.

        All tags with string values are decoded using the ISO-8859-1 character
        set. The resulting strings may contain characters that are legal in the
```

```
      character set but are illegal in XML documents. Illegal XML characters
      are replaced with the space (0x20) character.

    Supported datasets
      The following datasets from the application record are extracted.

        2:00  recordVersion
        2:05  objectName
        2:07  editStatus
        2:10  urgency
        2:15  category
        2:20  supplementalCategory
        2:22  fixtureIdentifer
        2:25  keyword
        2:26  contentLocation:code
        2:27  contentLocation:name
        2:40  instructions
        2:55  dateCreated
        2:60  timeCreated
        2:62  digitalCreationDate
        2:63  digitalCreationTime
        2:80  byline:author
        2:85  byline:authorTitle
        2:90  city
        2:92  subLocation
        2:95  provinceState
        2:100 country
        2:101 location
        2:103 transmissionReference
        2:105 headline
        2:110 credit
        2:115 source
        2:116 copyright
        2:118 contact
        2:120 caption
        2:122 captionWriter
        2:135 languageId

    Structure
      The schema defines a number of types, both simple and complex,
      to represent some of the IPTC tags. The itpcMetadataType is defined
      as a sequence of all the supported IPTC tags. This type is used to define
      a single global element that can appear in an instance document.

  </xsd:documentation>
</xsd:annotation>

<!-- Basic type definitions  -->
<xsd:simpleType name="urgencyType">
  <xsd:restriction base="xsd:positiveInteger">
    <xsd:minExclusive value="1"/>
    <xsd:maxInclusive value="8"/>
  </xsd:restriction>
</xsd:simpleType>
<xsd:complexType name="locationType">
  <xsd:sequence>
    <xsd:element name="code" type="xsd:string"/>
    <xsd:element name="name" type="xsd:string"/>
  </xsd:sequence>
</xsd:complexType>
```

```
  <xsd:complexType name="bylineType">
    <xsd:sequence>
      <xsd:element name="author" type="xsd:string"/>
      <xsd:element name="authorTitle" minOccurs="0" type="xsd:string"/>
    </xsd:sequence>
  </xsd:complexType>

  <!-- Type definition for the global element  -->
  <xsd:complexType name="iptcMetadataType">
    <xsd:sequence>
      <xsd:element name="recordVersion" minOccurs="0" type="xsd:integer"/>
      <xsd:element name="objectName" minOccurs="0" type="xsd:string"/>
      <xsd:element name="editStatus" minOccurs="0" type="xsd:string"/>
      <xsd:element name="urgency" minOccurs="0" type="urgencyType"/>
      <xsd:element name="category" minOccurs="0" type="xsd:string"/>
      <xsd:element name="supplementalCategory" minOccurs="0" maxOccurs="unbounded"
           type="xsd:string"/>
      <xsd:element name="fixtureIdentifier" minOccurs="0" type="xsd:string"/>
      <xsd:element name="keyword" minOccurs="0" maxOccurs="unbounded"
           type="xsd:string"/>
      <xsd:element name="contentLocation" minOccurs="0" maxOccurs="unbounded"
           type="locationType"/>
      <xsd:element name="instructions" minOccurs="0" type="xsd:string"/>
      <xsd:element name="dateCreated" minOccurs="0" type="xsd:date"/>
      <xsd:element name="timeCreated" minOccurs="0" type="xsd:string"/>
      <xsd:element name="digitalCreationDate" minOccurs="0"   type="xsd:date"/>
      <xsd:element name="digitalCreationTime" minOccurs="0"  type="xsd:string"/>
      <xsd:element name="byline" minOccurs="0" maxOccurs="unbounded"
           type="bylineType"/>
      <xsd:element name="city" minOccurs="0" type="xsd:string"/>
      <xsd:element name="subLocation" minOccurs="0" type="xsd:string"/>
      <xsd:element name="provinceState" minOccurs="0" type="xsd:string"/>
      <xsd:element name="country" minOccurs="0" type="xsd:string"/>
      <xsd:element name="location" minOccurs="0" type="xsd:string"/>
      <xsd:element name="transmissionReference" minOccurs="0" type="xsd:string"/>
      <xsd:element name="headline" minOccurs="0" type="xsd:string"/>
      <xsd:element name="credit" minOccurs="0" type="xsd:string"/>
      <xsd:element name="source" minOccurs="0" type="xsd:string"/>
      <xsd:element name="copyright" minOccurs="0" type="xsd:string"/>
      <xsd:element name="contact" minOccurs="0" maxOccurs="unbounded"
           type="xsd:string"/>
      <xsd:element name="caption" minOccurs="0" type="xsd:string"/>
      <xsd:element name="captionWriter" minOccurs="0" maxOccurs="unbounded"
           type="xsd:string"/>
      <xsd:element name="languageId" minOccurs="0" type="xsd:string"/>
    </xsd:sequence>
  </xsd:complexType>

  <!-- The gobal element -->
  <xsd:element name="iptcMetadata" type="iptcMetadataType"/>

</xsd:schema>
```

## F.3 XML Schema for ORDImage Attributes

This schema is the content model for the object attributes of ORDImage. The namespace for this schema is `http://xmlns.oracle.com/ord/meta/ordimage`.

```
<?xml version="1.0" encoding="UTF-8"?>
<!-- Copyright 2004, 2005 Oracle.  All rights reserved. -->
```

```
<xsd:schema xmlns="http://xmlns.oracle.com/ord/meta/ordimage"
     targetNamespace="http://xmlns.oracle.com/ord/meta/ordimage"
     xmlns:xsd="http://www.w3.org/2001/XMLSchema"
     elementFormDefault="qualified" attributeFormDefault="unqualified">

  <xsd:annotation>
    <xsd:documentation>

      Introduction
        Oracle Multimedia schema for ORDSYS.ORDImage attributes
        Elements are optional and can appear in any order.
    </xsd:documentation>
  </xsd:annotation>

  <xsd:complexType name="ordImageAttributesType">
    <xsd:all minOccurs="0">
      <xsd:element name="height" type="xsd:positiveInteger" minOccurs="0"/>
      <xsd:element name="width" type="xsd:positiveInteger" minOccurs="0"/>
      <xsd:element name="contentLength" type="xsd:positiveInteger" minOccurs="0"/>
      <xsd:element name="fileFormat" type="xsd:string" minOccurs="0"/>
      <xsd:element name="contentFormat" type="xsd:string" minOccurs="0"/>
      <xsd:element name="compressionFormat" type="xsd:string" minOccurs="0"/>
      <xsd:element name="mimeType" type="xsd:string" minOccurs="0"/>
    </xsd:all>
  </xsd:complexType>

  <xsd:element name="ordImageAttributes" type="ordImageAttributesType"/>
</xsd:schema>
```

## F.4 XML Schema for XMP Metadata

This schema is the content model for XMP metadata retrieved from images. It is also the content model for writing metadata to images. The namespace for this schema is `http://xmlns.oracle.com/ord/meta/xmp`.

```
<?xml version="1.0" encoding="UTF-8"?>
<!-- Copyright (c) 2004, 2005, Oracle.  All rights reserved. -->
<xsd:schema xmlns="http://xmlns.oracle.com/ord/meta/xmp"
     xmlns:xsd="http://www.w3.org/2001/XMLSchema"
     xmlns:xdb="http://xmlns.oracle.com/xdb"
     targetNamespace="http://xmlns.oracle.com/ord/meta/xmp"
     elementFormDefault="qualified"
     attributeFormDefault="unqualified">

  <xsd:annotation>
    <xsd:documentation>

      Introduction
       This is the Oracle Multimedia schema for metadata embedded in XMP packets.
        The schema provides for a single element from the RDF namespace.
        As defined in the January 2004 version of the XMP specification,
        this element should be an rdf:RDF element.
        XMP is defined by Adobe Systems Incorporated. For more information
        about XMP, see the XMP Specification at the Adobe Web site.
    </xsd:documentation>
  </xsd:annotation>

  <xsd:complexType name="xmpMetadataType" mixed="false">
    <xsd:sequence>
      <xsd:any namespace="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
```

```
            xdb:SQLType="CLOB"
            processContents="skip" minOccurs="1" maxOccurs="1"/>
      </xsd:sequence>
    </xsd:complexType>
    <xsd:element name="xmpMetadata" type="xmpMetadataType"/>
</xsd:schema>
```

# G

# Exceptions for Oracle Multimedia Objects

This appendix describes these Oracle Multimedia object exceptions:



**See Also:**

*Oracle Database Error Messages* for information about Oracle Multimedia error messages

## G.1 ORDAudioExceptions Exceptions

These 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 Multimedia 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.

**ORDAudioExceptions.NULL_SOURCE**

**Cause:** This exception is raised when the value of the ORDAudio.source attribute is NULL.

**Action:** Use an ORDAudio object that was created with the ORDAudio.init method (recommended). Or, set the ORDAudio.source attribute to an ORDSource object that you initialized.

## G.2 ORDDocExceptions Exceptions

These 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 Multimedia documentation for more information.

**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.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.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.

**ORDDocExceptions.NULL_SOURCE**

**Cause:** This exception is raised when the value of the ORDDoc.source attribute is NULL.

**Action:** Use an ORDDoc object that was created with the ORDDoc.init method (recommended). Or, set the ORDDoc.source attribute to an ORDSource object that you initialized.

## G.3 ORDImageExceptions Exceptions

These 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.

**ORDImageExceptions.NULL_SOURCE**

**Cause:** This exception is raised when the value of the ORDImage.source attribute is NULL.

**Action:** Use an ORDImage object that was created with the ORDImage.init method (recommended). Or, set the ORDImage.source attribute to an ORDSource object that you initialized.

## G.4 ORDImageSIExceptions Exceptions

These 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 Multimedia documentation for more information.

## G.5 ORDSourceExceptions Exceptions

These 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 Oracle Multimedia 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 Multimedia documentation for more information.

## G.6 ORDVideoExceptions Exceptions

These 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.METHOD_NOT_SUPPORTED**

**Cause:** This exception is raised when the method called is not supported.

**Action:** Do not call this method.

**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.NULL_SOURCE**

**Cause:** This exception is raised when the value of the ORDVideo.source attribute is NULL.

**Action:** Use an ORDVideo object that was created with the ORDVideo.init method (recommended). Or, set the ORDVideo.source attribute to an ORDSource object that you initialized.

**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 Multimedia documentation for more information.

# H

# init( ) Methods for Oracle Multimedia Object Types

This appendix presents reference information about the init( ) methods for these four Oracle Multimedia object types:

- ORDAudio
- ORDDoc
- ORDImage
- ORDVideo

In this appendix, references to ORD*xxx* objects represent these four Oracle Multimedia object types.

> **Note:** In Oracle Database 11g Release 2 (11.2), Oracle introduces new constructor functions to construct ORD*xxx* objects from BLOBs and other sources. The init( ) methods will continue to be available in Oracle Database 11*g* Release 2. However, Oracle recommends writing new applications to use the new constructor functions. Oracle also recommends migrating existing applications from the init( ) methods to the new constructor functions at your earliest convenience.
>
> See Chapter 3, Chapter 4, Chapter 5, and Chapter 6 for information about the new constructor functions for ORD*xxx* objects.

This appendix describes these methods:

# 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 these exceptions:

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

## Examples

None.

# 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 these exceptions:

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

## Examples

None.

# 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 these exceptions:

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

## Examples

None.

# 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 these exceptions:

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

## Examples

None.

# 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. (See Table 2–1.)

**srcLocation**
The source location of the audio data. (See Table 2–2.)

**srcName**
The source name of the audio data. (See Table 2–3.)

## Pragmas

None.

## Exceptions

None.

## Usage Notes

This constructor is a static method that initializes all the ORDAudio attributes to NULL with these 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

## Examples

None.

# 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. (See Table 2–1.)

**srcLocation**
The source location of the media data. (See Table 2–2.)

**srcName**
The source name of the media data. (See Table 2–3.)

## Pragmas

None.

## Exceptions

None.

## Usage Notes

This constructor is a static method that initializes all the ORDDoc attributes to NULL with these 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

## Examples

None.

# 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. (See Table 2–1.)

**srcLocation**
The source location of the image data. (See Table 2–2.)

**srcName**
The source name of the image data. (See Table 2–3.)

## Pragmas

None.

## Exceptions

None.

## Usage Notes

This constructor is a static method that initializes all the ORDImage attributes to NULL with these 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

## Examples

None.

# 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. (See Table 2–1.)

**srcLocation**
The source location of the video data. (See Table 2–2.)

**srcName**
The source name of the video data. (See Table 2–3.)

## Pragmas

None.

## Exceptions

None.

## Usage Notes

This constructor is a static method that initializes all the ORDVideo attributes to NULL with these 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

## Examples

None.

# Oracle Multimedia SQL/MM Still Image Object Types

Oracle Multimedia 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 I-5

  Describes the average color feature of an image.

- SI_Color Object Type on page I-11

  Encapsulates color values of a digitized image.

- SI_ColorHistogram Object Type on page I-16

  Describes the relative frequencies of the colors exhibited by samples of an image.

- SI_FeatureList Object Type on page I-24

  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 I-43

  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 I-48

  Represents digital images with inherent image characteristics such as height, width, format, and so on.

- SI_Texture Object Type on page I-69

  Describes the texture feature of the image characterized by the size of repeating items (coarseness), brightness variations (contrast), and predominant direction (directionality).

The StillImage object types are defined in the `ordisits.sql` file. After installation, this file is available in the Oracle home directory at:

*<ORACLE_HOME>*`/ord/im/admin` (on Linux and UNIX)

*<ORACLE_HOME>*`\ord\im\admin` (on Windows)

A public synonym with the corresponding object type name is created for each of these StillImage object types. Therefore, you need not specify the ORDSYS schema name when specifying a StillImage object type.

This appendix also includes these topics:

- SQL Functions and Procedures on page I-3

  Provides an overview of how the SQL functions and procedures are presented in this guide, and how they are created.

- Internal Helper Types on page I-4

  Provides syntax for attributes that are VARRAY types.

- Views on page I-74

  Describes the views in the SI_INFORMTN_SCHEMA that you can query for information about the supported image formats and implementation-defined values.

**See Also:**

*Oracle Multimedia 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's rights. A public synonym with the corresponding function or procedure name is created for all SQL functions and procedures. Therefore, you need not 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.

## 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 as follows:

- 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 I-16.

- 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 I-16.

- 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 I-43.

- 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 I-69.

## 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's 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 attributes for this object type are defined as follows in the `ordisits.sql` file:

```
-------------------
-- TYPE ATTRIBUTES
-------------------
SI_AverageColorSpec SI_Color,
```

where:

- SI_AverageColorSpec: the average color of the object.

## SI_AverageColor Constructors

This section describes these SI_AverageColor object constructors:

## 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 these conditions is true:

- The value of the specified averageColorSpec is NULL.
- The value of the specified averageColorSpec is not a valid SI_Color value.

### Examples

None.

# 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 these conditions is 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

None.

# 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 these conditions are 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

None.

## 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's 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 attributes for this object type are defined as follows in the `ordisits.sql` file:

```
-------------------
-- TYPE ATTRIBUTES
-------------------
redValue   INTEGER,
greenValue INTEGER,
blueValue  INTEGER,
```

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, you must use the system default constructor.
  - The default constructor does not perform any argument validation. By calling the SI_RGBColor method, specified values are 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

None.

# 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's rights. It is declared as an INSTANTIABLE and NOT FINAL type. This object type is defined as follows. (See Internal Helper Types 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 attributes for this object type are defined as follows in the `ordisits.sql` file:

```
-------------------
-- TYPE ATTRIBUTES
-------------------
SI_ColorsList       colorsList,
SI_FrequenciesList  colorFrequenciesList,
```

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 these SI_ColorHistogram object constructors:

# SI_ColorHistogram(colors, frequencies)

## Format

```
SI_ColorHistogram(SI_ColorsList        IN colorsList,
                  SI_FrequenciesList  IN colorFrequenciesList)
RETURN SELF AS RESULT DETERMINISTIC;
```

## Description

Constructs an SI_ColorHistogram object. These 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 for the SI_ColorsList and colorFrequenciesList attribute syntax.

## Pragmas

None.

## Format of Equivalent SQL Function

```
SI_ArrayClrHstgr(colors IN SI_ColorsList,
                 frequencies IN colorFrequenciesList),
RETURN SI_ColorHistogram DETERMINISTIC;
```

## 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.

## Exceptions

None.

## Usage Notes

An error is returned if one of these conditions is true:

- One of the specified values is NULL.
- One of the specified frequency values is less than 0 or greater than 100.

## Examples

None.

## 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. These 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 these conditions are true:

- One of the specified values is NULL.
- The frequency specified is less than 0 or greater than 100.

### Examples

None.

# 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. These 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 these conditions are 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 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

None.

## SI_ColorHistogram Methods

This section presents reference information on these SI_ColorHistogram methods, which are used for color histogram construction and image matching:

# 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 one of these conditions is true:

- Any of the specified values is NULL.
- The frequency is less than 0 or greater than 100.
- The attribute SI_ColorsList has SI_MaxHistogramLength elements.

## Pragmas

None.

## Exceptions

None.

## Examples

None.

## 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 one of these conditions 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

None.

# 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's 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 attributes for this object type are defined as follows in the `ordisits.sql` file:

```
-------------------
-- TYPE 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,
```

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 I-26

# 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 is not to 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 is not to 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 is not to 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 is not to be considered during image matching.

## Pragmas

None.

## Exceptions

None.

## Usage Notes

An error is returned if any of these conditions are 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

None.

# SI_FeatureList Methods

This section presents reference information on these SI_FeatureList methods, which are used for image matching:

## 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

None.

# 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

None.

## 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

None.

# 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

None.

## 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

None.

# 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

None.

## 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 is 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 these conditions are 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

None.

## 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

None.

# 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

None.

## 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

None.

# 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

None.

## 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

None.

# 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

None.

## 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's rights. It is declared as an INSTANTIABLE and NOT FINAL type. (See Internal Helper Types 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 attributes for this object type are defined as follows in the `ordisits.sql` file:

```
-------------------
-- TYPE ATTRIBUTES
-------------------
SI_ColorPositions  colorPositions,
```

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* equals 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 these conditions are 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 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

None.

# 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*) equals 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 one of these conditions 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

None.

# 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's 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.

The attributes for this object type are defined as follows in the `ordisits.sql` file:

```
-------------------
-- 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,
```

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 to extract and cache image features.
- 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 these SI_StillImage object constructors:

- SI_StillImage(content) on page I-51
- SI_StillImage(content, explicitFormat) on page I-52
- SI_StillImage(content, explicitFormat, height, width) on page I-54

  This is an Oracle extension to the SQL/MM Still Image standard.

> **Note:** To construct SI_StillImage objects, Oracle 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 G for more information about this exception.

### Usage Notes

None.

### Examples

None.

# 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 Oracle Multimedia 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.

See Appendix G for more information about this exception.

## Usage Notes

An error is returned if the explicitFormat parameter is a NULL value, or if either of these 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 that combination of values results 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

None.

# 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 for Oracle Multimedia to use if the image is in an unsupported format.

**height**
The value for the height_SI attribute for Oracle Multimedia to use if the image is in an unsupported format.

**width**
The value for the width_SI attribute for Oracle Multimedia to use if the image is in an unsupported format.

## Pragmas

None.

## Exceptions

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.

ORDImageSIExceptions.NULL_CONTENT

This exception is raised if the content parameter is NULL.

See Appendix G for more information about these exceptions.

## Usage Notes

An error message is returned if the explicitFormat parameter value is a NULL value, or if either of these 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 that combination of values results 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

None.

# SI_StillImage Methods

This section presents reference information on these SI_StillImage methods, which are used for image data manipulation:

## 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

None.

# SI_InitFeatures( )

## Format

```
SI_InitFeatures( );
```

## Description

Extracts the image features and caches them in the SI_StillImage object. This method must be called once, after which SI_StillImage manages the image features such that every time the image is processed, new image features are automatically 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 are slower.
  - Image matching methods such as SI_Score are 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.

See Appendix G for more information about this exception.

## Examples

None.

## 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 these conditions are 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 Multimedia 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

None.

# 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

None.

## 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

None.

# 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

None.

## 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

None.

# SI_RetainFeatures( )

## Format

SI_RetainFeatures( );

RETURN BOOLEAN DETERMINISTIC;

## Description

Returns a Boolean value (TRUE or FALSE) to indicate whether to extract and cache image features.

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

None.

## 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 these 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.

See Appendix G for more information about this exception.

### Examples

None.

# 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 is 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

None.

## 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 for Oracle Multimedia to use for the thumbnail image.

**image**
The image for which you want to generate a thumbnail image.

**width**
The width for Oracle Multimedia 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 must be one fourth of the original image, then the height argument must be 100 divided by 2 and the width argument must 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

None.

# 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

None.

# 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's rights. It is declared as an INSTANTIABLE and NOT FINAL type. (See Internal Helper Types 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 attributes for this object type are defined as follows in the `ordisits.sql` file:

```
-------------------
-- TYPE ATTRIBUTES
-------------------
SI_TextureEncoding  textureEncoding,
```

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 I-71

# 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 these conditions are 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

None.

# 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 one of these conditions 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

None.

# 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 I–1 describes the SI_IMAGE_FORMATS view. This view identifies the supported image formats.

***Table I–1 SI_IMAGE_FORMATS View***

| Column Name | Data Type | Description |
|---|---|---|
| SI_FORMAT | VARCHAR2(SI_MaxFormatLength) | A list of the supported image formats. |

Table I–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 I–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 I–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 I–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 these:<br>■ SI_AverageColor<br>■ SI_Texture<br>■ SI_PositionalColor<br>■ SI_ColorHistogram |

Table I–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 I–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 I–5 describes the SI_VALUES view. This view identifies the implementation-defined values.

***Table I–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 these SI_VALUE column values:<br>■ SI_MaxContentLength is the maximum length for the binary representation of the SI_StillImage attribute.<br>■ SI_MaxFeatureNameLength is the maximum length for the character representation of a basic image feature name.<br>■ SI_MaxFormatLength is the maximum length for the character representation of an image format indication.<br>■ SI_MaxHistogramLength is the maximum number of color/frequency pairs that are admissible in an SI_ColorHistogram feature value.<br>■ SI_MaxRGBColor is the maximum value for each component of a color value that is represented by the RGB color space.<br>■ SI_MaxTextureLength is the number of bytes needed for the encoded representation of an SI_Texture object.<br>■ SI_MaxValueLength is the maximum length for the character representation (name) of the meta-variables in the SI_VALUES view.<br>■ 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 these values:<br>■ 0<br>If the implementation places no limit on the meta-variable defined by SI_VALUE column or cannot determine the limit.<br>■ NULL<br>If the implementation does not support any features for which the meta-variable is applicable.<br>■ Any non-NULL, nonzero value<br>The maximum size supported by the implementation for this meta-variable. |

# J

# Oracle Multimedia Deprecated API Components

This appendix lists deprecated components and components that will not be enhanced. This appendix includes these sections:



For detailed information about deprecated Oracle Multimedia API components and other features, see the Oracle Multimedia documentation in the Oracle Database Online Documentation Library. Specifically, see the documentation for releases earlier than the release when the component was deprecated.

## J.1 Image Processing Operators

The image processing operators dither and interleave were deprecated in Oracle Database 11*g* Release 1 (11.1). Although these operators still function, Oracle recommends that you stop using them as soon as possible. The dither operator has been replaced with the quantize operator. The interleave operator has been replaced with the contentFormat operator. See Chapter 5 and Appendix D for more information about the quantize and contentFormat operators.

## J.2 ORDImageSignature Methods

All ORDImageSignature methods were deprecated in Oracle Database 11*g* Release 1 (11.1).

## J.3 ORDImage Support for DICOM

In Oracle Database 11*g*, Release 1 (11.1), Oracle introduced new and substantially enhanced features to support DICOM content. As a result, beginning with Oracle Database 11*g*, Release 2 (11.2), the DICOM support in ORDImage objects that was introduced in Oracle Database 10*g*, Release 2 (10.2) is not being enhanced, and may be deprecated in a future release.

**Note:** The Oracle Database 10*g*, Release 2 DICOM support is still available in Oracle Database 11*g*, Release 2. However, Oracle recommends writing new medical imaging applications to use the DICOM support that was introduced in Oracle Database 11*g*, Release 1. Oracle also recommends migrating existing applications from the DICOM support in Oracle Database 10*g*, Release 2 (ORDImage objects) to the DICOM support in Oracle Database 11*g*, Release 1 at your convenience. See "Migrating from Release 10.2 DICOM Support" in *Oracle Multimedia DICOM Developer's Guide* for instructions.

**See Also:**

*Oracle Multimedia DICOM Developer's Guide* for more information about Oracle Multimedia DICOM features and enhancements

# Index

## Numerics



## A



## B



## C

## D



## E



## F



## G

## H



## I



## J



## L



## M

## N



## O

## P

## Q



## R



## S

## T



## U



## V



## W



## X



## Y