Oracle8*i*™ *inter*Media Audio, Image, and Video

User's Guide and Reference

Release 8.1.5

February 1999

Part No. A67299-01

Oracle8*i* *inter*Media audio, image, and video is a component of Oracle8*i* *inter*Media, a product designed to manage multimedia Web content within Oracle8*i*.

ORACLE®

Enabling the Information Age™

Oracle8*i* *inter*Media Audio, Image, and Video User's Guide and Reference

Part No. A67299-01

Release 8.1.5

Primary Authors: Rod Ward and Max Chittister

Contributors: Alok Srivastava, Raja Chatterjee, Dan Mullen, Susan Mavris, Todd Rowell, Ashok Joshi, Susan Kotsovo, Rosanne Toohey, Bill Beauregard, Jeff Hebert, Chuck Murray, Susan Shepard, Brenda Silva, Deborah Laderoute

Contents

3 ORDAudio Reference Information

4 ORDImage Reference Information

6 ORDSource Reference Information

D Image Raw Pixel Format

E Sample Programs

F Reference Information

List of Examples

List of Tables

Send Us Your Comments

Oracle8*i interM*edia Audio, Image, and Video User's Guide and Reference, Release 8.1.5

Part No. A67299-01

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

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

If you find any errors or have any other suggestions for improvement, please indicate the chapter, section, and page number (if available).

You can send comments to us in the following ways:

- e-mail: nedc_doc@us.oracle.com
- FAX - 603.897.3316. Attn: Oracle8*i interM*edia Documentation
- postal service:
  Oracle Corporation
  Oracle8*i interM*edia Documentation
  One Oracle Drive
  Nashua, NH 03062
  USA

If you would like a reply, please include your name, and a postal or e-mail address.

Preface

This guide describes how to use Oracle8*i* *inter*Media audio, image and video.

Oracle8*i* *inter*Media audio, image, and video requires Oracle8*i* or Oracle8*i* Enterprise Edition.

For information about the differences between Oracle8*i* and Oracle8*i* Enterprise Edition and the features and options that are available to you, see *Getting to Know Oracle8i*.

Intended Audience

This guide is intended for application developers and database administrators who are interested in storing, retrieving, and manipulating audio, image, and video data in an Oracle database, including developers of audio, image, and video specialization options.

Structure

This guide contains the following chapters and appendixes:

Chapter 1	Introduces multimedia and Oracle8*i* *inter*Media; explains multimedia-related concepts.
Chapter 2	Provides basic examples of using Oracle8*i* *inter*Media object types and methods.
Chapter 3	Provides reference information on Oracle8*i* *inter*Media ORDAudio object type and methods.
Chapter 4	Provides reference information on Oracle8*i* *inter*Media ORDImage object type and methods.

Chapter 5	Provides reference information on Oracle8*i* *inter*Media ORDVideo object type and methods.
Chapter 6	Provides reference information on Oracle8*i* *inter*Media ORDSource object type and methods.
Chapter 7	Provides tuning tips for the DBA for more efficient storage of multimedia data.
Appendix A	Describes the supported audio data formats.
Appendix B	Describes the supported image data formats.
Appendix C	Describes the process and processCopy Operators.
Appendix D	Describes the raw pixel format.
Appendix E	Describes how to run the sample application and includes a source listing of that program.
Appendix F	Provides reference information for object types and methods described in the demonstration programs.
Appendix G	Emphasizes several entries from the online FAQ.
Appendix H	Lists exceptions raised and potential errors, their causes, and user actions to correct them.
Appendix I	Describes the deprecated image object types and methods.
Glossary	Defines important terms related to data options and multimedia information.

Related Documents

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

ORACLE_HOME`/ord/img/admin/README.txt`

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

For more information about using *inter*Media in a development environment, see the following documents in the Release 8.1.5 Oracle database server documentation set:

- *Oracle Call Interface Programmer's Guide*
- *Oracle8i Application Developer's Guide - Fundamentals*
- *Oracle8i Application Developer's Guide - Large Objects (LOBs)*
- *Oracle8i Concepts*
- *PL/SQL User's Guide and Reference*
- *Oracle8i interMedia Audio, Image, and Video Java Client User's Guide and Reference*

Conventions

In this guide, Oracle8*i* *inter*Media is sometimes referred to as *inter*Media.

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

The following conventions are also used in this guide:

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

1

Introduction

Oracle8*i* *inter*Media Audio, Image and Video is a component of the Oracle8*i* *inter*Media product. Oracle8*i* *inter*Media Audio, Image and Video enables Oracle8*i* to manage image, audio, and video in an integrated fashion with other enterprise information, through object types. Oracle8*i* *inter*Media extends Oracle8*i* reliability, availability, and data management to multimedia content in Internet, electronic commerce, and media-rich applications.

The capabilities of *inter*Media audio, image, and video encompass the storage, retrieval, management, and manipulation of multimedia data managed by Oracle8*i*. *inter*Media audio, image, and video support multimedia storage, retrieval, and management of:

- Binary large objects (BLOBs) stored locally in Oracle8*i* and containing audio, image, or video data
- File-based large objects, or BFILEs, stored locally in operating system-specific file systems and containing audio, image, or video data
- URLs containing audio, image, or video data stored on any HTTP server such as the Oracle Application Server, Netscape Application Server, Microsoft Internet Information Server, Apache HTTPD server, and Spyglass servers
- Streaming audio or video data stored on specialized media servers such as the Oracle Video Server, Real Audio Server, and Real Video Server
- Any other specially formatted multimedia data stored in BLOBs and any user-defined sources on other servers

*inter*Media is designed to be extensible. It supports a base set of popular audio, image, and video data characteristics for multimedia processing that also can be extended, for example, to support additional formats, new digital compression and

decompression schemes (**codecs**), data sources, and even specialized data processing algorithms for audio and video data.

*inter*Media is a building block for various multimedia applications rather than being an end-user application. It consists of object types along with related methods for managing and processing multimedia data. Some example applications for *inter*Media audio, image, and video are:

- Internet music stores that provide music samplings of CD quality
- Digital sound repositories
- Dictation and telephone conversation repositories
- Audio archives and collections (for example, for musicians)
- Digital art galleries
- Real estate marketing
- Document imaging
- Photograph collections (for example, for professional photographers)
- Internet video stores and digital video-clip previews
- Digital video sources for streaming video delivery systems
- Digital video libraries, archives, and repositories
- Libraries of digital video training programs
- Digital video repositories (for example, for motion picture production, television broadcasting, documentaries, advertisements, and so forth)

These applications have common and unique requirements. Oracle8*i* *inter*Media Audio, Image, and Video object types support common application requirements and can be extended to address application-specific requirements. With Oracle8*i* *inter*Media, multimedia data can be managed as easily as standard attribute data.

Oracle8*i* *inter*Media is accessible to applications through both relational and object interfaces. Database applications written in JAVA, C++, or traditional 3GLs can interface to *inter*Media through modern class library interfaces, or PL/SQL and the Oracle Call Interface (OCI).

*inter*Media supports storage of many of the popular file formats, including desktop publishing image and streaming audio and video formats in Oracle8*i* databases. *inter*Media provides the means to add audio, image, and video columns or objects to existing tables, insert and retrieve multimedia data, and provide limited conversion between application formats. This enables database designers to extend exist-

ing application databases with multimedia data or to build new end-user multimedia database applications. *inter*Media developers can use the basic functions provided here to build specialized multimedia applications.

1.1 Oracle8*i* *inter*Media

Oracle8*i* *inter*Media uses object types, similar to JAVA or C++ classes, to describe audio, image, and video data. These object types are called ORDAudio, ORDImage, and ORDVideo. An instance of these object types consists of attributes, including metadata and the media data, and methods. Media data is the actual audio, image, or video. Metadata is information about the data, including information like object length, compression, or format. Methods are procedures that can be performed on the object like getContent( ) and process( ).

*inter*Media objects (ORDAudio, ORDImage, and ORDVideo) have a common media data storage model. The media data component of these objects can be stored in the database, in a binary large object (BLOB) under transaction control. The media data can also be stored outside the database without transaction control. In this case, a pointer is stored in the database under transaction control and the media data is stored in:

- An external binary file (BFILE)
- An HTTP server-based URL
- A source on a specialized media data server like the Oracle Video Server, the Real Networks streaming server, or Live Pictures Image Server
- A user-defined source on other servers

Media data stored outside the database can provide a convenient mechanism for managing large, pre-existing or new media repositories that reside as flat files on erasable or read-only media. This data can be imported into BLOBs at any time for transaction control.

Object metadata and methods are always stored in the database under Oracle8*i* *inter*Media control. Whether media data is stored within or outside the database, *inter*Media manages metadata for all the media types and may automatically extract it for audio, image, and video. This metadata includes the following attributes:

- Local (audio, image, and video) data in the database
- Audio, image, and video data storage information including the source type, location, and source name, and whether the data is stored locally (in the database) or externally

- Audio, image, and video data update time stamp
- Audio and video data description
- Audio, image, and video data format
- MIME type of the audio, image, and video data
- Audio and video comments
- Audio characteristics: encoding type, number of channels, sampling rate, sample size, compression type, and play time (duration)
- Image characteristics: height and width, image content length, image content format, and image compression format
- Video characteristics: frame width and height, frame resolution, frame rate, play time (duration), number of frames, compression type, number of colors, and bit rate

In addition, a minimal set of image and data manipulation methods is provided. For image, this includes verifying the image properties match the image, performing format conversion and compression, scaling, cropping, copying, and deleting images.

It is possible to extend Oracle8*i* *inter*Media by:

- Creating a new object type or a new composite object type based on the provided multimedia object types. See the examples in Section 2.1.15 and Section 2.3.13 for more information.
- Creating specialized plug-ins to support other external sources of audio, image, and video data that are not currently supported. See Section 1.6.1 for more information.
- Creating specialized audio and video data format plug-ins to support other audio and video data formats that are not currently supported. See Section 1.6.1 for more information.
- Using the setProperties( ) method for foreign images, which allows certain other image formats to be recognized. See Section 1.6.1 and the "setProperties( ) Method for Foreign Images" in Chapter 4 for more information.
- Using the audio and video data processing methods to allow a specific audio or video command and its arguments to be passed through to process audio or video data. See Section 1.6.2 and Section 1.6.3 for more information.

1.2 Audio Concepts

This section contains information about digitized audio concepts and using *inter*Media audio to build audio applications or specialized *inter*Media audio objects.

1.2.1 Digitized Audio

Audio may be produced by an audio recorder, an audio source such as a microphone, digitized audio, other specialized audio recording devices, or even by program algorithms. Audio recording devices take an analog or continuous signal, such as the sound picked up by a microphone or sound recorded on magnetic media, and convert it into digital values with specific audio characteristics such as format, encoding type, number of channels, sampling rate, sample size, compression type, and audio duration.

*inter*Media audio integrates the storage, retrieval, and management of digitized audio data in Oracle databases using Oracle8*i*.

*inter*Media audio supports applications that either play or process audio data that are in a particular file format with a specific sampling rate, encoding type, sample size, and number of channels depending upon available hardware capabilities or processing power for user-defined formats. *inter*Media audio is extensible so that it can support any variety of special audio characteristics.

1.2.2 Audio Components

Digitized audio consists of the audio data (digitized bits) and attributes that describe and characterize the audio data. Audio applications sometimes associate application-specific information, such as the description of the audio clip, date recorded, author or artist, and so forth, with audio data by storing descriptive text in an attribute or column in the database table.

The audio data can have different formats, encoding types, compression types, numbers of channels, sampling rates, sample sizes, and playing times (duration) depending upon how the audio data was digitally recorded. Each audio data characteristic is crucial to audio data access and represents the audio data quality.

Given that an audio clip may not have any compression, description, and source information (source type, source location, source name), the minimal attributes carried along with an audio clip may include the file format, MIME type, encoding type, number of channels, sampling rate, sample size, and audio duration. These data attributes describe or characterize the audio data as it was recorded or produced by the digitized recording device.

The size of digitized audio (number of bytes) tends to be large compared to traditional computer objects, such as numbers and text. Therefore, several encoding schemes are used that squeeze audio data into fewer bytes, thus putting a smaller load on storage devices and networks.

1.3 Image Concepts

This section contains information about digitized image concepts and using *inter*Media image to build image applications or specialized *inter*Media image objects.

1.3.1 Digitized Images

*inter*Media image can help integrate the storage, retrieval, and management of digitized images in Oracle databases using Oracle8*i*.

*inter*Media image supports two-dimensional, static, digitized images stored as binary representations of real-world objects or scenes. Images may be produced by a document or photograph scanner, a video source such as a camera or VCR connected to a video digitizer or frame grabber, other specialized image capture devices, or even by program algorithms. Capture devices take an analog or continuous signal such as the light that falls onto the film in a camera, and convert it into digital values on a two-dimensional grid of data points known as pixels. Devices involved in the capture and display of images are under application control.

1.3.2 Image Components

Digitized images consist of the image data (digitized bits) and attributes that describe and characterize the image data. Image applications sometimes associate application-specific information, such as including the name of the person pictured in a photograph, description of the image, date photographed, photographer, and so forth, with image data by storing this descriptive text in an attribute or column in the database table.

The image data (pixels) can have varying depths (bits per pixel) depending on how the image was captured, and can be organized in various ways. The organization of the image data is known as the **data format**. *inter*Media image can read and write image data using a variety of current data formats. See Appendix B for a list of supported data formats. In addition, certain foreign images (formats not natively supported by *inter*Media image) have limited support. See Appendix D for more information.

Given that an image may not have any compression and source information (source type, source location, source name), the minimal attributes of an image may include

such things as its size (height in scan lines and width in pixels), and the file and content format.

The storage space required for digitized images can be large compared to traditional attribute data such as numbers and text. Many compression schemes are available to squeeze an image into fewer bytes, thus reducing storage device and network load. **Lossless compression** schemes squeeze an image so that when it is decompressed, the resulting image is bit-for-bit identical with the original. **Lossy compression** schemes do not result in an identical image when decompressed, but rather, one in which the changes may be imperceptible to the human eye.

Image **interchange format** describes a well-defined organization and use of image attributes, data, and often compression schemes, allowing different applications to create, exchange, and use images. Interchange formats are often stored in or as disk files. They may also be exchanged in a sequential fashion over a network and be referred to as a **protocol**. There are many application subdomains within the digitized imaging world and many applications that create or utilize digitized images within these. *inter*Media image supports reading and writing many interchange formats (see Appendix B).

1.4 Video Concepts

This section contains information about digitized video concepts and using *inter*Media video to build video applications or specialized *inter*Media video objects.

1.4.1 Digitized Video

Video may be produced by a video recorder, a video camera, digitized animation video, other specialized video recording devices, or even by program algorithms. Some video recording devices take an analog or continuous signal, such as the video picked up by a video camera or video recorded on magnetic media, and convert it into digital values with specific video characteristics such as format, encoding type, frame rate, frame size, frame resolution, video length, compression type, number of colors, and bit rate.

*inter*Media video can help integrate the storage, retrieval, and management of digitized video data in Oracle databases using Oracle8*i*.

*inter*Media video supports applications that either play or process video data that are in a particular file format. This file format has a specific frame rate, frame size, frame resolution, compression type, video length, bit rate, and number of colors depending upon available hardware capabilities or processing power for user-

defined formats. *inter*Media video is extensible so that it can support any variety of special video characteristics.

1.4.2 Video Components

Digitized video consists of the video data (digitized bits) and the attributes that describe and characterize the video data. Video applications sometimes associate application-specific information, such as the description of the video training tape, date recorded, instructor's name, producer's name, and so forth, with video data by storing descriptive text in an attribute or column in the database table.

The video data can have different formats, compression types, frame rates, frame sizes, frame resolutions, playing times, compression types, number of colors, and bit rates depending upon how the video data was digitally recorded. Each video data characteristic is crucial to video data access and represents the video data quality.

Given that a video clip may not have any compression, description, and source information (source type, source location, source name), the minimal attributes carried along with a video clip may include the file format, MIME type, encoding type, frame rate, frame size, frame resolution, total length of playing time, total number of frames, number of colors, and bit rate. These data attributes describe the video data as it was recorded or produced by the digitized recording device.

The size of digitized video (number of bytes) tends to be large compared to traditional computer objects, such as numbers and text. Therefore, several encoding schemes are used that squeeze video data into fewer bytes, thus putting a smaller load on storage devices and networks.

1.5 Object Relational Technology

Oracle8*i* is an *object relational* database management system. That means that in addition to its traditional role in the safe and efficient management of relational data, it now provides support for the definition of object types, including the data involved in objects and the operations (methods) that can be performed on them. This powerful mechanism, well established in the object-oriented world, includes integral support for BLOBs to provide the basis for adding complex objects, such as digitized audio, image, and video to Oracle8*i* databases.

Within Oracle8*i* *inter*Media, audio data characteristics have an object relational type known as **ORDAudio,** image data characteristics have an object relational type known as **ORDImage,** and video data characteristics have an object relational type known as **ORDVideo**. All three store data source information in an object relational type known as **ORDSource**.

See the following references for extensive information on using BLOBs and BFILEs:

- *Oracle8i Application Developer's Guide - Large Objects (LOBs)*
- *Oracle8i Concepts*

  See the chapter on Object Views.

1.5.1 Multimedia Object Types and Methods

Oracle8*i* *inter*Media provides the ORDAudio, ORDImage, and ORDVideo object types and methods for:

- Performing updateTime ORDSource attribute manipulation
- Performing description attribute manipulation (*inter*Media audio and video only)
- Manipulating multimedia data source attribute information
- Getting and managing multimedia data from Oracle8*i* *inter*Media, Web servers, and other servers
- Performing file operations (open, close, trim, read, and write) on the source (*inter*Media audio and video only)
- Performing comments attribute manipulation (*inter*Media audio and video only)
- Extracting attributes from multimedia data
- Performing minimal manipulation operations on multimedia data (*inter*Media image only)
- Processing commands (processAudioCommand and processVideoCommand) to operate on the multimedia data (*inter*Media audio and video only)

1.5.2 ORDSource Object Type and Methods

Oracle8*i* *inter*Media provides the ORDSource object type and methods for multimedia data source manipulation. This section presents a conceptual overview of the ORDSource object type methods.

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

1.5.2.1 Storing Multimedia Data

*inter*Media can store multimedia data as an internal source within the Oracle8*i* database, under transactional control as a BLOB. It can also externally reference digitized multimedia data stored as an external source in an operating system-specific BFILE in a local file-system, as a URL on an HTTP server, as streaming audio or video stored on media servers, or as a user-defined source on other servers. Although these external storage mechanisms are particularly convenient for integrating pre-existing sets of multimedia data with an Oracle8*i* database, the multimedia data will not be under transactional control.

BLOBs are stored in the database tablespaces in a way that optimizes space and provides efficient access. BLOBs may not be stored inline with other row data. Depending on the size of the BLOB, a locator is stored in the row and the actual BLOB (up to 4 gigabytes) is stored in other tablespaces. The locator can be considered a pointer to the actual location of the BLOB value. When you select a BLOB, you are selecting the locator instead of the value, although this is done transparently. An advantage of this design is that multiple BLOB locators can exist in a single row. For example, you might want to store a short video clip of a training tape, an audio recording containing a brief description of its contents, a syllabus of the course, a picture of the instructor, and a set of maps and directions to each training center.

Because BFILEs are not under the transactional control of the database, users could change the external source without updating the database, thus causing an inconsistency with the BFILE locator. Only the locator for a BFILE, URL, streaming audio or video source on a media server, or a user-defined source located on other servers is stored in the row. See *Oracle8i Application Developer's Guide - Large Objects (LOBs)* and *Oracle Call Interface Programmer's Guide* for detailed information on using BLOBs and BFILEs.

*inter*Media provides the ORDSource object type and methods for performing local attribute manipulation, updateTime attribute manipulation, source attribute manipulation, import/export operations, source content operations, source access operations, source read and write operations, and command process operations.

1.5.2.2 Querying Multimedia Data

Once stored within an Oracle8*i* database, multimedia data can be queried and retrieved by using the various alphanumeric columns (attributes) of the table to find a row that contains the data. For example, you can select a video clip from the Training table where the course name is 'Oracle8*i* Concepts'.

The collection of multimedia data in the database can be related to some set of attributes or keywords that describe the associated content. The multimedia data content can be described with textual components and numeric attributes such as

dates and identification numbers. For Oracle8*i*, data attributes can now reside in the same table as the object type. Alternatively, the application designer could define a composite object type that contains one of the object types along with other attributes.

1.5.2.3 Accessing Multimedia Data

Applications access and manipulate multimedia data using SQL, PL/SQL, or JAVA through the object relational types ORDAudio, ORDImage, and ORDVideo. See *Oracle8i interMedia Audio, Image, and Video Java Client User's Guide and Reference* for more information about using JAVA. The object syntax for accessing attributes within a complex object is the dot notation:

```
variable.data_attribute
```

The syntax for invoking methods of a complex object is also the dot notation:

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

See *Oracle8i Concepts* for information on this and other SQL syntax.

1.6 Extending Oracle8*i* *inter*Media

*inter*Media audio, *inter*Media image, and *inter*Media video can be extended to support:

- Other external sources of audio, image, and video data not currently supported
- Other audio, image, and video data formats not currently supported
- Audio and video data processing

The following sections describe each of these topics and where to find more information.

1.6.1 Supporting Other External Sources and Other Audio, Image, and Video Data Formats

For each unique external audio, image, or video data source or each unique audio or video data format that you want to support, you must:

1. Design your new data source or new audio or video data format.
2. Implement your new data source or new audio or video data format.
3. Install your new plug-in in the ORDPLUGINS schema.
4. Grant EXECUTE privileges on your new plug-in to PUBLIC.

Supporting Other External Sources

To implement your new data source, you must implement the required interfaces in the ORDX_<srcType>_SOURCE package in ORDPLUGINS schema (where <srcType> represents the name of the new external source type). Use the package body example in Section 6.3.2 as a template to create the package body. Then set the source parameter in the setSourceInformation( ) call to the appropriate source value to indicate to the audio, image, or video object that package ORDPLUGINS.ORDX_<srcType>_SOURCE is available as a plug-in. Use the ORDPLUGINS.ORDX_FILE_SOURCE and ORDPLUGINS.ORDX_HTTP_SOURCE packages as guides when you extend support to other external audio, image, or video sources.

See Section 2.4, Section 6.3.1, Section 6.3.2, and Section 6.3.4 for examples and for more information on extending the supported external sources of audio, image, and video data.

Supporting Other Audio and Video Data Formats

To implement your new audio or video data format, you must implement the required interfaces in the ORDPLUGINS.ORDX_<format>_<media> package in the ORDPLUGINS schema (where <format> represents the name of the new audio or video data format and <media> represents the media of the format). Use the ORDPLUGINS.ORDX_DEFAULT_<media> package as a guide when you extend support to other audio or video data formats. Use the package body examples in Section 3.3.6 and Section 5.3.6 as templates to create the audio or video package body, respectively. Then set the new format parameter in the setFormat( ) call to the appropriate format value to indicate to the audio or video object that package ORDPLUGINS.ORDX_<format>_<media> is available as a plug-in.

See Section E.1 and Section E.3 for more information on installing your own format plug-in and running the sample scripts provided.

See Section 2.1.14, Section 2.3.12, Section 3.3.1, and Section 5.3.1 for examples and for more information on extending the supported audio and video data attributes.

Supporting Other Image Data Formats

Oracle8*i* *inter*Media image supports certain other image formats through the setProperties( ) method for foreign images. This method allows other image formats to be recognized by writing the values supplied to the setProperties( ) method for foreign images to the existing ORDImage data attributes. See “setProperties( ) Method for Foreign Images” in Chapter 4 for more information.

1.6.2 Supporting Audio Data Processing

To support audio data processing, that is, the passing of an audio processing command and set of arguments to a format plug-in for processing, use the processAudioCommand( ) method. This method is only available for user-defined formats.

See "processAudioCommand( ) Method" in Chapter 3 and Section 2.1.14 for a description.

1.6.3 Supporting Video Data Processing

To support video data processing, that is, the passing of a command and set of arguments to a format plug-in for processing, use the processVideoCommand( ) method. This method is only available for user-defined formats.

See "processVideoCommand( ) Method" in Chapter 5 and Section 2.3.12 for a description.

2

*inter*Media Examples

This chapter provides examples that show common operations with Oracle8*i* *inter*Media. Examples are presented by audio, image, and video data groups followed by a section that describes how to extend *inter*Media to support a new data source.

2.1 Audio Data Examples

*inter*Media audio examples include the following common operations:

- Defining a song object named songObject
- Creating an object table named SongsTable
- Creating a list object named songList that contains a list of songs
- Defining the implementation of the songList object
- Creating a CD object and CdTable table
- Inserting a song into the SongsTable table
- Inserting a CD into the CdTable table
- Loading a song into the SongsTable table
- Inserting a reference to a song object into the songs list in the CdTable table
- Adding a CD reference to a song
- Adding comments about a song
- Retrieving audio data from a song in a CD
- Displaying all the comments for a song in a CD
- Extending *inter*Media to support a new audio data format
- Extending *inter*Media audio with new object types

- Using *inter*Media with object views

The audio data examples in this section use a table of songs and a table of CDs. For each song, the following information is stored: a CDRef (REF into the CD table), a song ID, song title, artist, awards, time period of song, duration of song, clipRef (REF into the audio clips table or music video), text content, and the audio source containing lyrics. (A REF refers to row objects with globally unique object IDs that capture references between row objects; row objects are automatically indexed for fast access.) For each CD the following are stored: an item ID, CD DB ID, CD title, CD artist, CD category, copyright, name of producer, awards, time period, rating, duration, text content, cover image, and the list of songs on the CD.

Reference information on the methods used in these examples is presented in Chapter 3.

2.1.1 Defining a Song Object

Example 2–1 describes how to define a Song object.

Example 2–1 Define a Song Object

```
CREATE TYPE songObject as OBJECT (
  cdRef       REF CdObject,   -- REF into the cd table
  songId      VARCHAR2(20),
  title       VARCHAR2(4000),
  artist      VARCHAR2(4000),
  awards      VARCHAR2(4000),
  timePeriod  VARCHAR2(20),
  duration    INTEGER,
  clipRef     REF clipObject, -- REF into the clips table (music video)
  txtcontent  CLOB,
  audioSource ORDSYS.ORDAUDIO
);
```

2.1.2 Creating an Object Table SongsTable

Example 2–2 describes how to create an object table named SongsTable.

Example 2–2 Create a Table Named SongsTable

```
CREATE TABLE SongsTable of songObject (UNIQUE (songId), songId NOT NULL);
```

2.1.3 Creating a List Object Containing a List of References to Songs

Example 2–3 describes how to create a list object containing a list of references to songs.

Example 2–3 Create a List Object Containing a List of References to Songs

```
CREATE TYPE songNstType AS TABLE of REF songObject;

CREATE TYPE songList AS OBJECT (songs songNstType,
     MEMBER PROCEDURE addSong(s IN REF songObject));
```

2.1.4 Defining the Implementation of the songList Object

Example 2–4 describes how to define the implementation of the songList object.

Example 2–4 Define the Implementation of the songList Object

```
CREATE TYPE BODY songList AS
  MEMBER PROCEDURE addSong(s IN REF songObject)
  IS
    pos INTEGER := 0;
  BEGIN
    IF songs IS NULL THEN
      songs := songNstType(NULL);
      pos := 0;
    ELSE
      pos := songs.count;
    END IF;
    songs.EXTEND;
    songs(pos+1) := s;
  END;
END;
```

2.1.5 Creating a CD Object and a CD Table

This section describes how to create a CD object and a CD table of audio clips that includes, for each audio clip, the following information:

- Item ID
- CD DB ID
- CD title

- CD artist
- CD category
- Copyright
- Name of producer
- Awards
- Time period
- Rating
- Duration
- Text content
- Cover image
- Songs

Example 2–5 creates a CD object named CdObject, and a CD table named CdTable that contains the CD information.

Example 2–5 Create a CD Table Containing CD Information

```
CREATE TYPE CdObject as OBJECT (
  itemId        INTEGER,
  cddbID        INTEGER,
  title         VARCHAR2(4000),
  artist        VARCHAR2(4000),
  category      VARCHAR2(20),
  copyright     VARCHAR2(4000),
  producer      VARCHAR2(4000),
  awards        VARCHAR2(4000),
  timePeriod    VARCHAR2(20),
  rating        VARCHAR2(256),
  duration      INTEGER,
  txtcontent    CLOB,
  coverImg      REF ORDSYS.ORDImage,
  songs         songList);

CREATE TABLE CdTable OF CdObject (UNIQUE(itemId), itemId NOT NULL)
        NESTED TABLE songs.songs STORE AS song_store_table;
```

2.1.6 Inserting a Song into the SongsTable Table

Example 2–6 describes how to insert a song into the SongsTable table.

Example 2–6 Insert a Song into the SongsTable Table

```
-- insert a song into the songs table
INSERT INTO SongsTable VALUES (NULL,
                               '00',
                               'Under Pressure',
                               'Queen',
                               'no awards',
                               '80-90',
                               243,
                               NULL,
                               EMPTY_CLOB(),
                               ORDSYS.ORDAudio(NULL,
ORDSYS.ORDSource(EMPTY_BLOB(),NULL,NULL,NULL,NULL,NULL),
NULL, NULL, EMPTY_CLOB(), NULL, NULL, NULL, NULL, NULL, NULL));

-- check songs insertion
SELECT s.title
FROM   SongsTable s
WHERE songId = '00';
```

2.1.7 Inserting a CD into the CdTable Table

Example 2–7 describes how to insert a CD into the CdTable table.

Example 2–7 Insert a CD into the CdTable Table

```
-- insert a cd into the cd table
INSERT INTO CdTable VALUES (1, 23232323,
                            'Queen Classics',
                            'Queen',
                            'rock',
                            'BMV Company',
                            'BMV',
                            'Grammy',
                            '80-90',
                            'no rating',
                            4000,             -- in seconds
                            EMPTY_CLOB(),
                            NULL,
                            songList(NULL));

-- check cd insertion
SELECT cd.title
FROM   Cdtable cd;
```

2.1.8 Loading a Song into the SongsTable Table

Example 2–8 describes how to load a song into the SongsTable table. This example requires an AUDDIR directory to be defined; see the comments in the example.

Example 2–8 Load a Song into the SongsTable Table

```
-- Load a Song into the SongsTable
-- Create your directory specification below
-- CREATE OR REPLACE DIRECTORY AUDDIR AS '/audio/';
DECLARE
        audioObj ORDSYS.ORDAUDIO;
        ctx RAW(4000) :=NULL;
BEGIN
        SELECT S.audioSource INTO audioObj
        FROM   SongsTable S
        WHERE  S.songId = '00'
        FOR UPDATE;

        audioObj.setSource('FILE', 'AUDDIR', 'UnderPressure.au');
        audioObj.setMimeType('audio/basic');
        audioObj.import(ctx);
        audioObj.setProperties(ctx);

        UPDATE SongsTable S
        SET    S.audioSource = audioObj
        WHERE  S.songId = '00';
        COMMIT;
END;

-- check song insertion
DECLARE
     audioObj ORDSYS.ORDAUDIO;
     ctx RAW(4000) :=NULL;
BEGIN
     SELECT S.audioSource INTO audioObj
     FROM   SongsTable S
     WHERE  S.songId = '00';

     dbms_output.put_line('Content Length: ' ||
                    audioObj.getContentLength(ctx));
     dbms_output.put_line('Content MimeType: ' ||
                    audioObj.getMimeType());
END;
```

2.1.9 Inserting a Reference to a Song Object into the Songs List in the CdTable Table

Example 2–9 describes how to insert a reference to a song object into the songs list in the CdTable table.

Example 2–9 Insert a Reference to a Song Object into the Songs List in the CdTable Table

```
-- Insert a reference to a SongObject into the Songs List in the CdTable Table
DECLARE
        songRef REF SongObject;
        songListInstance songList;
BEGIN
        SELECT REF(S) into songRef
        FROM   SongsTable S
        where  S.songId = '00';

        SELECT C.songs INTO songListInstance
        FROM   CdTable C
        WHERE  C.itemId = 1
        FOR UPDATE;

        songListInstance.addSong(songRef);

        UPDATE CdTable C
        SET    C.songs = songListInstance
        WHERE  C.itemId = 1;

        COMMIT;
END;

-- check insertion of ref
-- this example works for the first entry inserted in the songList
DECLARE
        song             SongObject;
        songRef          REF SongObject;
        songListInstance songList;
        songType         songNstType;
BEGIN
        SELECT C.songs INTO songListInstance
        FROM   CdTable C
        WHERE  C.itemId = 1;

        SELECT songListInstance.songs INTO songType FROM DUAL;
        songRef := songType(1);
```

```
        SELECT DEREF(songRef) INTO song FROM DUAL;

        dbms_output.put_line('Song Title: ' ||
                             song.title);
END;
```

2.1.10 Adding a CD Reference to a Song

Example 2–10 describes how to add a CD reference to a song.

Example 2–10 Add a CD Reference to a Song

```
-- Adding a cd reference to a song
DECLARE
     songCdRef  REF CdObject;
BEGIN
     SELECT S.cdRef INTO songCdRef
     FROM   SongsTable S
     WHERE  S.songId = '00'
     FOR UPDATE;

     SELECT REF(C) INTO songCdRef
     FROM  CdTable C
     WHERE C.itemId = 1;

     UPDATE SongsTable S
     SET    S.cdRef = songCdRef
     WHERE  S.songId = '00';

     COMMIT;
END;

-- check cd Ref
DECLARE
     cdRef REF CdObject;
     cd    CdObject;
BEGIN
     SELECT S.cdRef INTO cdRef
     FROM   SongsTable S
     WHERE  S.songId = '00';

     SELECT DEREF(cdRef) INTO cd FROM DUAL;

     dbms_output.put_line('Cd Title: ' ||
                          cd.title);
```

```
END;
```

2.1.11 Adding Comments About a Song

Example 2–11 describes how to add comments about a song.

Example 2–11 Add Comments About a Song

```
-- Adding comments about a song
DECLARE
        audioObj ORDSYS.ORDAUDIO;
        comments VARCHAR2(256);
BEGIN
        SELECT S.audioSource into audioObj
        FROM   SongsTable S
        WHERE  S.songId = '00'
        FOR UPDATE;

        comments := 'I like this song when I am Under Pressure';
        audioObj.writeToComments(1, 41, comments);

        UPDATE SongsTable S
        SET    S.audioSource = audioObj
        WHERE  S.songId = '00';

        COMMIT;
END;
```

2.1.12 Retrieving Audio Data from a Song in a CD

Example 2–12 describes how to retrieve audio data from a song in a CD.

Example 2–12 Retrieve Audio Data from a Song in a CD

```
FUNCTION retrieveAudio(cdId IN INTEGER,
 songId IN INTEGER) RETURN BLOB IS obj ORDSYS.ORDAudio;
BEGIN
 select S.audioSource into obj from SongsTable S
   where S.songId = songId;
 return obj.getContent;
END;
```

2.1.13 Displaying All Comments for a Song in a CD

Example 2–13 describes how to display all comments for a song in a CD.

Example 2–13 Display All Comments for a Song in a CD

```
-- Displaying comments about a song
DECLARE
        audioObj ORDSYS.ORDAUDIO;
        comments VARCHAR2(256);
BEGIN
        SELECT S.audioSource into audioObj
        FROM   SongsTable S
        WHERE  S.songId = '00';

        comments := audioObj.readFromComments(1, 41);
        dbms_output.put_line('Comments: ' || comments);
END;
```

2.1.14 Extending *inter*Media to Support a New Audio Data Format

To support a new audio data format, implement the required interfaces in the ORDX_<format>_AUDIO package in the ORDPLUGINS schema (where <format> represents the name of the new audio data format). See Section 3.3.1 for a complete description of the interfaces for the ORDX_DEFAULT_AUDIO package. Use the package body example in Section 3.3.6 as a template to create the audio package body. Then set the new format parameter in the setFormat call to the appropriate format value to indicate to the audio object that package ORDPLUGINS.ORDX_<format>_AUDIO is available as a plug-in.

See Section E.1 for more information on installing your own format plug-in and running the sample scripts provided. See the fplugins.sql and fpluginb.sql files that are installed in the *$ORACLE_HOME*/ord/aud/demo/ directory. These are demonstration (demo) plug-ins that you can use as a guideline to write any format plug-in that you want to support. See the auddemo.sql file in this same directory to learn how to install your own format plug-in.

2.1.15 Extending *inter*Media with a New Type

This section describes how to extend Oracle8*i* *inter*Media with a new object type.

You can use any of the *inter*Media objects types as the basis for a new type of your own creation.

See Example 2–3 and Example 2–4 for a brief example. See Example 2–26 for a more complete example and description.

2.1.16 Using Audio Types with Object Views

This section describes how to use audio types with object views. Just as a view is a virtual table, an object view is a virtual object table.

Oracle provides object views as an extension of the basic relational view mechanism. By using object views, you can create virtual object tables from data -- of either built-in or user-defined types -- stored in the columns of relational or object tables in the database.

Object views can offer specialized or restricted access to the data and objects in a database. For example, you might use an object view to provide a version of an employee object table that does not have attributes containing sensitive data or a deletion method. Object views also let you try object-oriented programming without permanently converting your tables. Using object views, you can convert data gradually and transparently from relational tables to object-relational tables.

Consider the following non-object audio table:

```
create table flat (
   id                NUMBER,
   description       VARCHAR2(4000),
   localData         BLOB,
   srcType           VARCHAR2(4000),
   srcLocation       VARCHAR2(4000),
   srcName           VARCHAR2(4000),
   upDateTime        DATE,
   local             NUMBER,
   format            VARCHAR2(31),
   mimeType          VARCHAR2(4000),
   comments          CLOB,
   encoding          VARCHAR2(256),
   numberOfChannels  NUMBER,
   samplingRate      NUMBER,
   sampleSize        NUMBER,
   compressionType   VARCHAR2(4000),
   audioDuration     NUMBER,
   audioclip         RAW(2000)
);
```

You can create an object view on the flat table as follows:

```
create or replace view object_audio_v as
  select
      id,
      ordsys.ORDAudio(
```

```
      T.description,
      T.localData,
      T.comments,
      T.format,
      T.encoding,
      T.numberOfChannels,
      T.samplingRate,
      T.sampleSize,
      T.compressionType,
      T.audioDuration,
      T.audioclip) AUDIO
from flat T;
```

Object views provide the flexibility of looking at the same relational or object data in more than one way. Therefore, you can use different in-memory object representations for different applications without changing the way you store the data in the database. See the *Oracle8i Concepts* manual for more information on defining, using, and updating object views.

2.2 Image Data Examples

*inter*Media image examples include the following common operations:

- Adding types to a new or existing table
- Inserting a row using BLOB images
- Populating a row using BLOB images
- Inserting a row using BFILE images
- Populating a row using BFILE images
- Querying a row
- Importing an image from an external file into the database
- Copying an image
- Converting an image format
- Copying and converting an image in one step
- Extending *inter*Media with new object types
- Using image types with object views
- Addressing National Language Support (NLS) issues

2.2.1 Adding Image Types to an Existing Table

Suppose you have an existing table named 'emp' with the following columns:

```
ename      VARCHAR2(50)
salary     NUMBER
job        VARCHAR2(50)
department INTEGER
```

To add a new column to the 'emp' table called 'photo' using the ORDImage type, issue the statement in Example 2–14.

Example 2–14 adds a new column of type ORDImage to the emp table.

Example 2–14 Adding a New Column of Type ORDImage to the emp Table

```
ALTER TABLE emp
ADD (photo ORDSYS.ORDImage);
```

2.2.2 Adding Image Types to a New Table

Suppose you are creating a new table called 'emp' with the following information:

- Employee name
- Salary
- Job title
- Department
- Badge photograph
- Large photograph

The column for the badge photograph (maybe a thumbnail image cropped and scaled from the large personnel photograph) uses the ORDImage type, and the column 'large_photo' also uses the ORDImage type. The statement in Example 2–15 creates the table and adds ORDImage types to the new table.

Example 2–15 Adding ORDImage Types to a New Table

```
CREATE TABLE emp (
  ename VARCHAR2(50),
  salary NUMBER,
  job VARCHAR2(50),
  department INTEGER,
  photo ORDSYS.ORDImage,
```

```
  large_photo ORDSYS.ORDImage);
```

2.2.3 Inserting a Row Using BLOB Images

To insert a row into a table that has storage for image content using the ORDImage type, you must populate the type with an initializer. Note that this is different from NULL. Attempting to use the ORDImage type with a NULL value results in an error.

Example 2–16 describes how to insert rows into the table using the ORDImage type. Assume you have a table 'emp' with the following columns:

```
ename      VARCHAR2(50)
salary     NUMBER
job        VARCHAR2(50)
department INTEGER
photo   ORDImage
```

If you are going to store image data in the database (in a binary large object (BLOB)), you must populate the ORDSource.localData attribute with a value and initialize storage for the localData attribute with an empty_blob( ) constructor. To insert a row into the table with empty data in the 'photo' column, issue the statement in Example 2–16.

Example 2–16 inserts a row into a table with empty data in the ORDImage type column.

Example 2–16 Insert a Row into a Table with Empty Data in the ORDImage Type Column

```
INSERT INTO emp VALUES (
  'John Doe', 24000, 'Technical Writer', 123,
  ORDSYS.ORDImage(ORDSYS.ORDSource(empty_blob(),NULL,NULL,NULL,SYSDATE,1),
                  NULL,NULL,NULL,NULL,NULL,NULL,NULL));
```

2.2.4 Populating a Row Using BLOB Images

Prior to updating a BLOB value, you must lock the row containing the BLOB locator. This is usually done using a SELECT FOR UPDATE statement in SQL and PL/SQL programs, or using an Oracle Call Interface (OCI) pin or lock function in OCI programs.

Example 2–17 populates a row with ORDImage BLOB data.

Example 2–17 Populate a Row with ORDImage BLOB Data

```
DECLARE
    -- applicaition variables
    Image ORDSYS.ORDImage;
BEGIN
    INSERT INTO emp VALUES (
      'John Doe', 24000, 'Technical Writer', 123,
      ORDSYS.ORDImage(ORDSYS.ORDSource(empty_blob(), NULL,NULL,NULL,SYSDATE,1),
                      NULL,NULL,NULL,NULL,NULL,NULL,NULL));
    -- select the newly inserted row for update
    SELECT photo INTO Image FROM emp
           WHERE ename = 'John Doe' for UPDATE;
    --BEGIN
    -- use the getContent method to get the LOB locator.
    -- populate the data with dbms lob calls or write an OCI program to
    -- fill in the image BLOB.
    --END;
    -- set property attributes for the image data
    Image.setProperties;
    UPDATE emp SET photo = Image WHERE ename = 'John Doe';
    -- continue processing
END;
```

An UPDATE statement is required to update the property attributes. If you do not use the setProperties( ) method and UPDATE statement now, you can still commit and the change to the image will be reflected in the BLOB attribute, but not in the properties. See *Oracle8i Application Developer's Guide - Large Objects (LOBs)* for more information on BLOBs.

2.2.5 Inserting a Row Using BFILE Images

To insert a row into a table that has storage for image content in external files using the ORDImage type, you must populate the type with an initializer. Note that this is different from NULL. Attempting to use the ORDImage type with a NULL value results in an error.

Example 2–18 describes how to insert rows into the table using the ORDImage type. Assume you have a table 'emp' with the following columns:

```
ename      VARCHAR2(50)
salary     NUMBER
job        VARCHAR2(50)
department  INTEGER
large_photo ORDImage
```

If you are going to use the ORDImage type column, you must first populate the column with a value. To populate the value of the ORDImage type column with an image stored externally in a file, you must populate the row with a file constructor.

Example 2–18 inserts a row into the table with an image called 'jdoe.gif' from the ORDIMGDIR directory:

Example 2–18 Insert a Row into a Table with an Image in the ORDImage Type Column

```
INSERT INTO emp VALUES (
  'John Doe', 24000, 'Technical Writer', 123,
  ORDSYS.ORDImage(ORDSYS.ORDSource(empty_blob(),'file','ORDIMGDIR',
                                   'jdoe.gif',SYSDATE,0),
                  NULL,NULL,NULL,NULL,NULL,NULL,NULL));
```

Note: In this release of Oracle8*i*, the content of the ORDImage BFILE type is read-only.

For a description of row insertion into an object type, see Chapter 4 and the *Oracle8i Application Developer's Guide - Large Objects (LOBs)* manual.

The sourceLocation argument 'ORDIMGDIR' is a directory referring to a file system directory. Note that the directory name must be in uppercase. The following sequence creates a directory named ORDIMGDIR:

```
connect internal
        -- make a directory referring to a file system directory
create directory ORDIMGDIR as '<MYIMAGEDIRECTORY>';
grant read on directory ORDIMGDIR to <user-or-role>;
```

where <MYIMAGEDIRECTORY> is the file system directory, and <user-or-role> is the specific user to whom to grant read access.

2.2.6 Populating a Row Using BFILE Images

Example 2–19 populates the row with ORDImage data stored externally in files.

Example 2–19 Populate a Row with ORDImage External File Data

```
DECLARE
    Image ORDSYS.ORDImage;
BEGIN
```

```
    INSERT INTO emp VALUES ('John Doe', 24000, 'Technical Writer', 123,
    ORDSYS.ORDImage(ORDSYS.ORDSource(empty_blob(),'file','ORDIMGDIR',
                                     'jdoe.gif',SYSDATE,0),
                    NULL,NULL,NULL,NULL,NULL,NULL,NULL));
    -- select the newly inserted row for update
    SELECT large_photo INTO Image FROM emp
           WHERE ename = 'John Doe' FOR UPDATE;
    -- set property attributes for the image data
    Image.setProperties;
    UPDATE emp SET large_photo = Image WHERE ename = 'John Doe';
    -- continue processing
END;
```

2.2.7 Querying a Row

Example 2–20 and Example 2–21 assume you have this table:

```
create table emp (
ename VARCHAR2(50),
salary NUMBER,
job VARCHAR2(50),
department INTEGER,
photo ORDSYS.ORDImage,
large_photo ORDSYS.ORDImage);
```

Example 2–20 queries a row that has ORDImage data.You must create a table alias (E in this example) when you refer to a type in a SELECT statement.

Example 2–20 Querying a Row That Has ORDImage Data of Any Content Length

```
SELECT ename, E.large_photo.getWidth()
  FROM emp E
  WHERE ename = 'John Doe' and
        E.large_photo.getWidth() > 32;
```

Example 2–21 queries a row that has ORDImage data.

Example 2–21 Querying a Row That Has ORDImage Data of a Specific Content Length

```
SELECT ename, E.large_photo.getCompressionFormat()
  FROM emp E
  WHERE ename = 'John Doe' and
        E.large_photo.getWidth() > 32 and
        E.large_photo.getContentLength() > 10000;
```

2.2.8 Importing an Image from an External File into the Database

To import an image from an external file into the database, use the ORDImage.import method. The program in Example 2–22 imports image data from an external file into the database. The source type, source location, and source name must be set prior to calling the import( ) method.

Example 2–22 Import an Image from an External File

```
DECLARE
    Image ORDSYS.ORDImage;
    ctx RAW(4000) :=NULL;
BEGIN
    SELECT large_photo
      INTO Image FROM emp
      WHERE ename = 'John Doe' FOR UPDATE;
    -- import the image into the database
    Image.import(ctx);
    UPDATE emp SET large_photo = IMAGE
           WHERE ename = 'John Doe';
END;
```

2.2.9 Copying an Image

To copy an image, use the ORDImage.copy method. The program in Example 2–23 copies image data.

Example 2–23 Copy an Image

```
DECLARE
    Image_1 ORDSYS.ORDImage;
    Image_2 ORDSYS.ORDImage;
BEGIN
    SELECT photo INTO Image_1
       FROM emp WHERE ename = 'John Doe';
    SELECT photo INTO Image_2
       FROM emp WHERE ename = 'Also John Doe' FOR UPDATE;
    -- copy the data from Image_1 to Image_2
    Image_1.copy(Image_2);
    -- continue processing
    UPDATE emp SET photo = Image_2
        WHERE ename = 'Also John Doe';
END;
```

2.2.10 Converting an Image Format

To convert the image data into a different format, use the process( ) method.

> **Note:** The process( ) method processes only into a BLOB, so the image data must be stored locally.

The program in Example 2–24 converts the image data to the TIFF file format.

Example 2–24 Convert an Image Format

```
DECLARE
    Image ORDSYS.ORDImage;
BEGIN
    SELECT photo INTO Image FROM emp
        WHERE ename = 'John Doe' FOR UPDATE;
    -- convert the image to TIFF (in place)
    Image.process('fileFormat=TIFF');
    UPDATE emp SET photo = Image WHERE ename = 'John Doe';
END;
```

2.2.11 Copying and Converting in One Step

To make a copy of the image and convert it into one step, use the processCopy( ) method.

> **Note:** The processCopy( ) method processes only into a BLOB, so the destination image must be set to local and the localData attribute in the source must be initialized.

The program in Example 2–25 creates a thumbnail image, converts the image data to the TIFF image file format, copies it to a BLOB, and leaves the original image intact.

Example 2–25 Copy and Convert an Image Format

```
DECLARE
    Image_1 ORDSYS.ORDImage;
    Image_2 ORDSYS.ORDImage;
BEGIN
    SELECT photo, large_photo
        INTO Image_2, Image_1
```

```
        FROM emp
        WHERE ename = 'John Doe' FOR UPDATE;
    -- convert the image to a tiff thumbnail and store the result in Image_2
    Image_1.processCopy('fileFormat=TIFF fixedScale=32 32', Image_2);
    -- continue processing
    UPDATE emp SET photo = Image_2 WHERE ename = 'John Doe';
END;
```

Changes made by the processCopy( ) method can be rolled back. This technique may be useful for a temporary format conversion.

2.2.12 Extending *inter*Media with a New Type

You can use the ORDImage type as shown in Example 2–26 as the basis for a new type of your own creation.

Example 2–26 Extend Oracle8i interMedia Image with a New Object Type

```
CREATE TYPE AnnotatedImage AS OBJECT
    ( image ORDSYS.ORDImage,
      description VARCHAR2(2000),
      MEMBER PROCEDURE SetProperties(SELF IN OUT AnnotatedImage),
      MEMBER PROCEDURE Copy(dest IN OUT AnnotatedImage),
      MEMBER PROCEDURE ProcessCopy(command IN VARCHAR2,
                                   dest IN OUT AnnotatedImage)
    );
/

CREATE TYPE BODY AnnotatedImage AS
  MEMBER PROCEDURE SetProperties(SELF IN OUT AnnotatedImage) IS
  BEGIN
    SELF.image.setProperties;
    SELF.description :=
        'This is an example of using Image object as a subtype';
  END SetProperties;
  MEMBER PROCEDURE Copy(dest IN OUT AnnotatedImage) IS
  BEGIN
    SELF.image.copy(dest.image);
    dest.description := SELF.description;
  END Copy;
  MEMBER PROCEDURE ProcessCopy(command IN VARCHAR2,
                               dest IN OUT AnnotatedImage) IS
  BEGIN
    SELF.Image.processCopy(command,dest.image);
    dest.description := SELF.description;
```

```
  END ProcessCopy;
END;
/
```

After creating the new type, you can use it as you would any other type. For example:

```
create or replace directory TEST_DIR as 'C:\TESTS';

CREATE TABLE my_example(id NUMBER, an_image AnnotatedImage);
INSERT INTO my_example VALUES (1,
    AnnotatedImage(
        ORDSYS.ORDImage(
            ORDSYS.ORDSource(empty_blob(),'file','ORDIMGDIR',
                             'jdoe.gif',SYSDATE,0),
            NULL,NULL,NULL,NULL,NULL,NULL,NULL),
        NULL));
COMMIT;
DECLARE
    myimage AnnotatedImage;
BEGIN
    SELECT an_image INTO myimage FROM my_example;
    myimage.SetProperties;
    DBMS_OUTPUT.PUT_LINE('This image has a description of ');
    DBMS_OUTPUT.PUT_LINE(myimage.description);
    UPDATE my_example SET an_image = myimage;
END;
/
```

2.2.13 Using Image Types with Object Views

Just as a view is a virtual table, an object view is a virtual object table.

Oracle provides object views as an extension of the basic relational view mechanism. By using object views, you can create virtual object tables from data-- of either built-in or user-defined types -- stored in the columns of relational or object tables in the database.

Object views can offer specialized or restricted access to the data and objects in a database. For example, you might use an object view to provide a version of an employee object table that does not have attributes containing sensitive data or a deletion method. Object views also let you try object-oriented programming without permanently converting your tables. Using object views, you can convert data gradually and transparently from relational tables to object-relational tables.

Consider the following non-object image table:

```
CREATE TABLE flat(
    id                NUMBER,
    localData         BLOB,
    srcType           VARCHAR2(4000),
    srcLocation       VARCHAR2(4000),
    srcName           VARCHAR2(4000),
    updateTime        DATE,
    local             NUMBER,
    height            INTEGER,
    width             INTEGER,
    contentLength     INTEGER,
    fileFormat        VARCHAR2(4000),
    contentFormat     VARCHAR2(4000),
    compressionFormat VARCHAR2(4000),
    mimeType          VARCHAR2(4000)
    );
```

Example 2–27 creates an object view on the flat table.

Example 2–27 Create an Object View on a Flat Table

```
CREATE OR REPLACE VIEW object_images_v AS
    SELECT
      id,
      ORDSYS.ORDImage(
        ORDSYS.ORDSource(
          T.localData,
          T.srcType,
          T.srcLocation,
          T.srcName,
          T.updateTime,
          T.local),
        T.height,
        T.width,
        T.contentLength,
        T.fileFormat,
        T.contentFormat,
        T.compressionFormat,
        T.mimeType
        ) IMAGE
      FROM flat T;
```

Object views provide the flexibility of looking at the same relational or object data in more than one way. Thus you can use different in-memory object representations for different applications without changing the way you store the data in the database. See the *Oracle8i Concepts* manual for more information on defining, using, and updating object views.

2.2.14 Addressing National Language Support (NLS) Issues

Example 2–28 shows how to use the processCopy( ) method with language settings that use the comma as the decimal point. For example, when the territory is FRANCE, the decimal point is expected to be a comma. Notice the ",75" specified as the scale factor. This application addresses National Language Support issues.

Example 2–28 Address a National Language Support Issue

```
ALTER SESSION SET NLS_LANGUAGE = FRENCH;
ALTER SESSION SET NLS_TERRITORY = FRANCE;
DECLARE
    myimage ORDSYS.ORDImage;
    mylargeimage ORDSYS.ORDImage;
BEGIN
    SELECT photo, large_photo INTO myimage, mylargeimage
      FROM emp FOR UPDATE;
    myimage.setProperties;
    myimage.ProcessCopy('scale=",75"', mylargeimage);
    UPDATE emp SET photo = myimage, large_photo = mylargeimage;
END;
/
```

2.3 Video Data Examples

*inter*Media video examples include the following common operations:

- Defining a clip object named clipObject
- Creating an object table named clipsTable
- Creating a list object named clipList that contains a list of clips
- Defining the implementation of the clipList object
- Creating a video object and videoTable table
- Inserting a video clip into the clipsTable table
- Inserting a row into the videoTable table

- Loading a video into the clipsTable table
- Inserting a reference to a clipObject into the clips list in the video table
- Inserting a reference to a video object into the clip
- Retrieving a video clip from the videoTable table
- Extending *inter*Media to support a new video data format
- Extending *inter*Media with new object types
- Using video types with object views

The video examples in this section use a table of video clips and a table of videos. For each video clip the following are stored: a videoRef (REF into the video table), clip ID, title, director, casting, category, copyright, producer, awards, time period, rating, duration, cdRef (REF into CdObject for sound tracks), text content (indexed by CONTEXT), cover image (REF into the image table), and video source. For each video the following are stored: an item ID, duration, text content (indexed by CONTEXT), cover image (REF into the image table), and a list of clips on the video.

Reference information on the methods used in these examples is presented in Chapter 5 and Appendix F.

2.3.1 Defining a Clip Object

Example 2–29 describes how to define a clip object.

Example 2–29 Define a Clip Object

```
CREATE TYPE clipObject as OBJECT (
  videoRef      REF VideoObject,       -- REF into the video table
  clipId        VARCHAR2(20),          -- Id inside of the clip table
  title         VARCHAR2(4000),
  director      VARCHAR2(4000),
  casting       ORDSYS.ORDAnnotations, -- Pairs of characters/actors
  category      VARCHAR2(20),
  copyright     VARCHAR2(4000),
  producer      VARCHAR2(4000),
  awards        VARCHAR2(4000),
  timePeriod    VARCHAR2(20),
  rating        VARCHAR2(256),
  duration      INTEGER,
  cdRef         REF CdObject,          -- REF into a CdObject(soundtrack)
  txtcontent    CLOB,
  coverImg      REF ORDSYS.ORDImage,   -- REF into the ImageTable
```

```
  videoSource   ORDSYS.ORDVideo);
```

See Appendix F for a description of the ORDAnnotations type and its methods.

2.3.2 Creating an Object Table clipsTable

Example 2–30 describes how to create an object table named clipsTable.

Example 2–30 Create a Table Named clipsTable

```
CREATE TABLE ClipsTable of clipObject (UNIQUE (clipId), clipId NOT NULL)
     NESTED TABLE casting.annotations STORE AS annot_store_table2;
```

2.3.3 Creating a List Object Containing a List of Clips

Example 2–31 describes how to create a list object containing a list of clips.

Example 2–31 Create a List Object Containing a List of Clips

```
CREATE TYPE clipNstType AS TABLE of REF clipObject;

CREATE TYPE clipList AS OBJECT (clips clipNstType,
        MEMBER PROCEDURE addClip(c IN REF clipObject));
```

2.3.4 Defining the Implementation of the clipList Object

Example 2–32 describes how to define the implementation of the clipList object.

Example 2–32 Define the Implementation of the clipList Object

```
CREATE TYPE BODY clipList AS
  MEMBER PROCEDURE addClip(c IN REF clipObject)
  IS
    pos INTEGER := 0;
  BEGIN
    IF clips IS NULL THEN
      clips := clipNstType(NULL);
      pos := 0;
    ELSE
      pos := clips.count;
    END IF;
    clips.EXTEND;
    clips(pos+1) := c;
  END;
END;
```

2.3.5 Creating a Video Object and a Video Table

This section describes how to create a video object and a video table of video clips that includes, for each video clip, the following information:

- Item ID
- Duration
- Text content
- Cover image
- Clips

Example 2–33 creates a video object named videoObject and a video table named videoTable that contains the video information.

Example 2–33 Create a Video Table Containing Video Information

```
CREATE TYPE VideoObject as OBJECT (
        itemId        INTEGER,
        duration      INTEGER,
        txtcontent    CLOB,
        coverImg      REF ORDSYS.ORDImage,
        clips         clipList);

CREATE TABLE VideoTable OF VideoObject (UNIQUE(itemId),itemId NOT NULL)
        NESTED TABLE clips.clips STORE AS clip_store_table;
```

2.3.6 Inserting a Video Clip into the clipsTable Table

Example 2–34 describes how to insert a video clip into the clipsTable table.

Example 2–34 Insert a Video Clip into the clipsTable Table

```
-- Insert a Video Clip into the ClipsTable
insert into ClipsTable values (NULL,
                        '11',
                        'Oracle Commercial',
                        'Larry Ellison',
                        ORDSYS.ORDAnnotations(NULL),
                        'commercial',
                        'Oracle Corporation',
                        '',
                        'no awards',
                        '90s'
                        'no rating',
```

```
 30,
 NULL,
 EMPTY_CLOB(),
 NULL,
 ORDSYS.ORDVIDEO('Oracle Commercial 1 Video Clip',
 ORDSYS.ORDSource(EMPTY_BLOB(),NULL,NULL,NULL,NULL,NULL),
'QuickTime File Format',
'video/quicktime',
 EMPTY_CLOB(),
 160, 120, 72, 15, 30, 450, 'Cinepak', 256, 15000));
```

See Appendix F for a description of the ORDAnnotations object type and methods.

2.3.7 Inserting a Row into the videoTable Table

Example 2–35 describes how to insert a row into the videoTable table.

Example 2–35 Insert a Row into the videoTable Table

```
-- Insert a row into the VideoTable
insert into VideoTable values (11,
                               30,
                               NULL,
                               NULL,
                               clipList(NULL));
```

2.3.8 Loading a Video into the clipsTable Table

Example 2–36 describes how to load a video into the clipsTable table. This example requires a VIDDIR directory to be defined; see the comments in the example.

Example 2–36 Load a Video into the clipsTable Table

```
-- Load a Video into a clip
-- Create your directory specification below
-- CREATE OR REPLACE DIRECTORY VIDDIR AS '/video/';
DECLARE
     videoObj ORDSYS.ORDVIDEO;
     ctx RAW(4000) :=NULL;
BEGIN
     SELECT C.videoSource INTO videoObj
     FROM   ClipsTable C
     WHERE  C.clipId = '11'
     FOR UPDATE;
```

```
     videoObj.setDescription('Under Pressure Video Clip');
     videoObj.setMimeType('video/quicktime');
     videoObj.setFormat('QuickTime File Format');
     videoObj.setFrameSize(160, 120);
     videoObj.setFrameResolution(72);
     videoObj.setFrameRate(15);
     videoObj.setVideoDuration(30);
     videoObj.setNumberOfFrames(450);
     videoObj.setCompressionType('Cinepak');
     videoObj.setNumberOfColors(256);
     videoObj.setSource('FILE', 'VIDDIR', 'UnderPressure.mov');
     videoObj.import(ctx);

     UPDATE ClipsTable C
        SET    C.videoSource = videoObj
     WHERE  C.clipId = '11';
     COMMIT;
END;

-- check video insertion
DECLARE
     videoObj ORDSYS.ORDVideo;
     ctx RAW(4000) :=NULL;
BEGIN
     SELECT C.videoSource INTO videoObj
     FROM   ClipsTable C
     WHERE  C.clipId = '11';

     dbms_output.put_line('Content Length: ' ||
                    videoObj.getContentLength(ctx));
     dbms_output.put_line('Content MimeType: ' ||
                    videoObj.getMimeType());
END;
```

2.3.9 Inserting a Reference to a Clip Object into the Clips List in the videoTable Table

Example 2–37 describes how to insert a reference to a clip object into the clips list in the videoTable table.

Example 2–37 Insert a Reference to a Clip Object into the Clips List in the videoTable Table

```
-- Insert a reference to a ClipObject into the Clips List in the VideoTable
DECLARE
     clipRef          REF ClipObject;
```

```
        clipListInstance clipList;
BEGIN
     SELECT REF(C) into clipRef
        FROM   ClipsTable C
     where  C.clipId = '11';

     SELECT V.clips INTO clipListInstance
        FROM   VideoTable V
     WHERE  V.itemId = 11
     FOR UPDATE;

     clipListInstance.addClip(clipRef);

     UPDATE VideoTable V
     SET    V.clips = clipListInstance
     WHERE  V.itemId = 11;

     COMMIT;
END;

-- check insertion of clip ref
DECLARE
     clip             ClipObject;
     clipRef          REF ClipObject;
     clipListInstance clipList;
     clipType         clipNstType;
BEGIN
     SELECT V.clips INTO clipListInstance
     FROM   VideoTable V
     WHERE  V.itemId = 11;

     SELECT clipListInstance.clips INTO clipType FROM DUAL;
     clipRef := clipType(1);
     SELECT DEREF(clipRef) INTO clip FROM DUAL;

     dbms_output.put_line('Clip Title: ' ||
                          clip.title);
END;
```

2.3.10 Inserting a Reference to a Video Object into the Clip

Example 2–38 describes how to insert a reference to a video object into the clip.

Example 2–38 Insert a Reference to a Video Object into the Clip

```
-- Insert a reference to a video object into the clip
DECLARE
     aVideoRef REF VideoObject;
BEGIN
-- make a VideoRef an obj to use for update
     SELECT Cp.videoRef INTO aVideoRef
     FROM   ClipsTable Cp
     WHERE  Cp.clipId = '11'
     FOR UPDATE;

-- change its value
     SELECT REF(V) INTO aVideoRef
     FROM   VideoTable V
     WHERE  V.itemId = 11;

-- update database
     UPDATE ClipsTable C
     SET    C.videoRef = aVideoRef
     WHERE  C.clipId = '11';

     COMMIT;
END;
```

2.3.11 Retrieving a Video Clip from the videoTable

Example 2–39 describes how to retrieve a video clip from the videoTable table and return it as a BLOB. The program segment performs these operations:

1. Defines the retrieveVideo( ) method to retrieve the video clip by its clipId as an ORDVideo BLOB.
2. Selects the desired video clip (where `C.clipId = clipId`) and returns it using the getContent method.

Example 2–39 Retrieve a Video Clip

```
FUNCTION retrieveVideo(clipId IN INTEGER)
RETURN BLOB IS
 obj ORDSYS.ORDVideo;

BEGIN
   -- Select the desired video clip from the ClipTable table.
   SELECT C.videoSource INTO obj from ClipTable C
          WHERE C.clipId = clipId;
```

```
   return obj.getContent;
END;
```

2.3.12 Extending *inter*Media to Support a New Video Data Format

This section describes how to extend Oracle8*i* *inter*Media to support a new video data format.

To support a new video data format, implement the required interfaces in the ORDX_<format>_VIDEO package in the ORDPLUGINS schema (where <format> represents the name of the new video data format). See Section 5.3.1 for a complete description of the interfaces for the ORDX_DEFAULT_VIDEO package. Use the package body example in Section 5.3.6 as a template to create the video package body.

Then set the new format parameter in the setFormat call to the appropriate format value to indicate to the video object that package ORDPLUGINS.ORDX_<format>_VIDEO is available as a plug-in.

See Section E.3 for more information on installing your own format plug-in and running the sample scripts provided. See the fplugins.sql and fpluginb.sql files that are installed in the *$ORACLE_HOME*/ord/vid/demo/ directory. These are demonstration (demo) plug-ins that you can use as a guideline to write any format plug-in that you want to support. See the viddemo.sql file in this same directory to learn how to install your own format plug-in.

2.3.13 Extending *inter*Media with a New Object Type

This section describes how to extend Oracle8*i* *inter*Media with a new object type.

You can use the ORDSource, ORDVideo, and ORDAnnotations types as the basis for a new type of your own creation.

See Example 2–31 and Example 2–32 for a brief example. See Example 2–26 for a more complete example and description.

2.3.14 Using Video Types with Object Views

This section describes how to use video types with object views. Just as a view is a virtual table, an object view is a virtual object table.

Oracle provides object views as an extension of the basic relational view mechanism. By using object views, you can create virtual object tables from data -- of either built-in or user-defined types -- stored in the columns of relational or object tables in the database.

Object views can offer specialized or restricted access to the data and objects in a database. For example, you might use an object view to provide a version of an employee object table that does not have attributes containing sensitive data or a deletion method. Object views also let you try object-oriented programming without permanently converting your tables. Using object views, you can convert data gradually and transparently from relational tables to object-relational tables.

Consider the following non-object video table:

```
create table flat (
   id                number,
   description       VARCHAR2(4000),
   localData         BLOB,
   srcType           VARCHAR2(4000),
   srcLocation       VARCHAR2(4000),
   srcName           VARCHAR2(4000),
   upDateTime        DATE,
   local             NUMBER,
   format            VARCHAR2(31),
   mimeType          VARCHAR2(4000),
   comments          CLOB,
   width             INTEGER,
   height            INTEGER,
   frameResolution   INTEGER,
   frameRate         INTEGER,
   videoDuration     INTEGER,
   numberOfFrames    INTEGER,
   compressionType   VARCHAR2(4000),
   numberOfColors    INTEGER,
   bitRate           INTEGER,
   videoclip         RAW(2000)
   );
```

You can create an object view on the flat table as follows:

```
create or replace view object_video_v as
  select
      id,
      ordsys.ORDVideo(
         T.description,
         T.localData,
         T.comments,
         T.format,
         T.width,
         T.height,
         T.frameResolution,
```

```
        T.frameRate,
        T.videoDuration,
        T.numberofFrames,
        T.compressionType,
        T.numberOfColors,
        T.bitRate,
        T.videoclip) VIDEO
from flat T;
```

Object views provide the flexibility of looking at the same relational or object data in more than one way. Therefore, you can use different in-memory object representations for different applications without changing the way you store the data in the database. See the *Oracle8i Concepts* manual for more information on defining, using, and updating object views.

2.4 Extending *inter*Media to Support a New Data Source

This section describes how to extend Oracle8*i* *inter*Media to support a new data source.

To support a new data source, implement the required interfaces in the ORDX_<srcType>_SOURCE package in the ORDPLUGINS schema (where <srcType> represents the name of the new external source type). See Section 6.3.1 and Section 6.3.2 for a complete description of the interfaces for the ORDX_FILE_SOURCE and ORDX_HTTP_SOURCE packages. See Section 6.3.4 for an example of modifying the package body listing that is provided. Then set the source type parameter in the setSourceInformation call to the appropriate source type to indicate to the video object that package ORDPLUGINS.ORDX_<srcType>_SOURCE is available as a plug-in.

3

ORDAudio Reference Information

Oracle8*i* *inter*Media contains information about the ORDAudio type:

- Object type -- see Section 3.1.
- Methods -- see Section 3.2.
- Packages or PL/SQL plug-ins -- see Section 3.3.

The examples in this chapter assume that the test audio table TAUD has been created and filled with data. This table was created using the SQL statements described in Section 3.2.1.

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

Methods invoked at the ORDSource level that are handed off to the source plug-in for processing have ctx (RAW(4000)) 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 should invoke the closeSource( ) method.

Methods invoked from a source plug-in call have the first argument as ctx (RAW(4000)).

Methods invoked at the ORDAudio level that are handed off to the format plug-in for processing have ctx (RAW(4000)) 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, not all source or format plug-ins will use the ctx argument, but if you code as previously described, your application should work with any current or future source or format plug-in.

3.1 Object Types

Oracle8*i* *inter*Media describes the ORDAudio object type, which supports the storage and management of audio data.

ORDAudio Object Type

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

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

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

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

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

```
                RETURN RAW,

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

MEMBER PROCEDURE  setLocal,
MEMBER PROCEDURE  clearLocal,

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

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

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

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

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

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

MEMBER FUNCTION getContent RETURN BLOB,
```

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

MEMBER PROCEDURE deleteContent,

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

-- Methods associated with the comments attribute
MEMBER PROCEDURE appendToComments(amount IN BINARY_INTEGER,
                                  buffer IN VARCHAR2),
MEMBER PROCEDURE writeToComments(offset IN INTEGER,
                                 amount IN BINARY_INTEGER,
                                 buffer IN VARCHAR2),
MEMBER FUNCTION readFromComments(offset IN INTEGER,
                                 amount IN BINARY_INTEGER := 32767)
               RETURN VARCHAR2,
PRAGMA RESTRICT_REFERENCES(readFromComments, WNDS, WNPS, RNDS, RNPS),

MEMBER FUNCTION locateInComments(pattern    IN VARCHAR2,
                                 offset     IN INTEGER := 1,
                                 occurrence IN INTEGER := 1)
                RETURN INTEGER,
MEMBER PROCEDURE trimComments(newlen IN INTEGER),
MEMBER PROCEDURE eraseFromComments(amount IN OUT NOCOPY INTEGER,
                                   offset IN INTEGER := 1),
MEMBER PROCEDURE deleteComments,
MEMBER PROCEDURE loadCommentsFromFile(fileobj  IN BFILE,
                                      amount   IN INTEGER,
                                      from_loc IN INTEGER := 1,
                                      to_loc   IN INTEGER := 1),
MEMBER PROCEDURE copyCommentsOut(dest     IN OUT NOCOPY CLOB,
                                 amount   IN INTEGER,
```

```
                                from_loc IN INTEGER := 1,
                                to_loc   IN INTEGER := 1),
MEMBER FUNCTION compareComments(
                   compare_with_lob        IN CLOB,
                   amount                  IN INTEGER := 4294967295,
                   starting_pos_in_comment IN INTEGER := 1,
                   starting_pos_in_compare IN INTEGER := 1)
                RETURN INTEGER,
PRAGMA RESTRICT_REFERENCES(compareComments, WNDS, WNPS, RNDS, RNPS),

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

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

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

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

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

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

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

MEMBER PROCEDURE setAudioDuration(knownAudioDuration IN INTEGER),
```

```
MEMBER FUNCTION getAudioDuration RETURN INTEGER,
PRAGMA RESTRICT_REFERENCES(getAudioDuration, WNDS, WNPS, RNDS, RNPS),
MEMBER FUNCTION getAudioDuration(ctx IN OUT RAW) RETURN INTEGER,

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

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

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

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

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

where:

- description: the description of the audio object.
- source: the ORDSource where the audio data is to be found.
- format: the format in which the audio data is stored.
- mimeType: the Mime type information.
- comments: the comment 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 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 duration of the audio data stored.

3.2 Methods

This section presents reference information on the Oracle8*i* *inter*Media methods used for audio data manipulation. These methods are described in the following groupings:

ORDAudio Methods Associated with the updateTime Attribute

- getUpdateTime: returns the time when the audio object was last updated.
- setUpdateTime( ): sets the update time for the audio object.

ORDAudio Methods Associated with the description Attribute

- setDescription( ): sets the description of the audio data.
- getDescription: returns the description of the audio data.

ORDAudio Methods Associated with mimeType Attribute

- setMimeType( ): sets the MIME type of the stored audio data.
- getMimeType: returns the MIME type of the stored audio data.

ORDAudio Methods Associated with the source Attribute

- processSourceCommand( ): sends a command and related arguments to the source plug-in.
- isLocal: returns TRUE if the data is stored locally in a BLOB or FALSE if the data is external.
- setLocal: sets a flag to indicate that the data is stored locally in a BLOB.
- clearLocal: clears the flag to indicate that the data is stored externally.
- setSource( ): sets the source information to where audio data is found.

- getSource: returns a formatted string containing complete information about the external data source formatted as a URL.
- getSourceType: returns the external source type of the audio data.
- getSourceLocation: returns the external source location of the audio data.
- getSourceName: returns the external source name of the audio data.
- import( ): transfers data from an external data source (specified by calling setSourceInformation( )) to the local source (localData) within an Oracle database, setting the value of the local attribute to "1", meaning local and updating the timestamp.
- importFrom( ): transfers data from the specified external data source (source type, location, name) to the local source (localData) within an Oracle database, setting the value of the local attribute to "1", meaning local and updating the timestamp.
- export( ): transfers data from a local source (localData) within an Oracle database to the specified external data source, setting the value of the local attribute to "0", meaning external, and storing source information in the source.

  Note: Sources natively supported by *inter*Media are not writable and therefore do not support the export method. User-defined sources may support the export method.

- getContentLength( ): returns the length of the data source (as number of bytes).
- getContentInLob( ): returns content into a temporary LOB.
- getContent: returns the handle to the BLOB used to store contents locally.
- deleteContent: deletes the content of the local BLOB.

ORDAudio Methods Associated with File Operations

- openSource( ): opens a data source or a BLOB.
- closeSource( ): closes a data source or a BLOB.
- trimSource( ): trims a data source or a BLOB.
- readFromSource( ): reads a buffer of n bytes from a source beginning at a start position.

- writeToSource( ): writes a buffer of n bytes to a source beginning at a start position.

ORDAudio Methods Associated with the comments Attribute

- appendToComments( ): appends a specified buffer and amount of comment data to the end of the audio data comments.
- writeToComments( ): writes a specified buffer and amount of comment data to the audio data comments beginning at the specified offset.
- readFromComments( ): reads a specified amount of comment data from the audio data comments beginning at the specified offset.
- locateInComments( ): matches and locates the occurrence of the specified pattern of characters in the audio data comments.
- trimComments( ): trims the audio data comments to the specified length.
- eraseFromComments( ): erases the specified amount of comment data from the audio data comments beginning at the specified offset.
- deleteComments: deletes the audio data comments.
- loadCommentsFromFile( ): loads the comments from the specified BFILE into the audio data comments.
- copyCommentsOut( ): copies the audio data comments to the given Character LOB (CLOB).
- compareComments( ): compares the audio data comments with the comments of another specified CLOB of audio data.
- getCommentLength: returns the length of the audio data comments.

ORDAudio Methods Associated with Audio Attributes Accessors

- setFormat( ): sets the object attribute value of the format of the audio data.
- getFormat: returns the object attribute value of the format in which the audio data is stored.
- getFormat( ): calls the format plug-in to read the actual format embedded in the stored audio data.
- setEncoding( ): sets the object attribute value of the encoding type of the audio data.

- getEncoding: returns the object attribute value of the encoding type of the audio data.
- getEncoding( ): calls the format plug-in to read the actual encoding type embedded in the stored audio data.
- setNumberOfChannels( ): sets the object attribute value of the number of audio channels of the audio data.
- getNumberOfChannels: returns the object attribute value of the number of audio channels in the audio data.
- getNumberOfChannels( ): calls the format plug-in to read the actual number of channels embedded in the stored audio data.
- setSamplingRate( ): sets the object attribute value of the sampling rate of the audio data.
- getSamplingRate: returns the object attribute value of the sampling rate in samples per second at which the audio data was recorded.
- getSamplingRate( ): calls the format plug-in to read the actual sampling rate embedded in the stored audio data.
- setSampleSize( ): sets the object attribute value of the sample width or number of samples of audio in the data.
- getSampleSize: returns the object attribute value of the sample width or number of samples of audio in the data.
- getSampleSize( ): calls the format plug-in to read the actual sample width or number of samples embedded in the stored audio data.
- setCompressionType( ): sets the object attribute value of the compression type of the audio data.
- getCompressionType: returns the object attribute value of the compression type of the audio data.
- getCompressionType: calls the format plug-in to read the actual compression type embedded in the stored audio data. This method is available only for user-defined format plug-ins.
- setAudioDuration( ): sets the object attribute value of the total time value for the time required to play the audio data.
- getAudioDuration: returns the object attribute value of the total time required to play the audio data.

- getAudioDuration( ): calls the format plug-in to read the actual playing time embedded in the stored audio data. This method is available only for user-defined format plug-ins.
- setKnownAttributes( ): sets known audio attributes including format, encoding type, number of channels, sampling rate, sample size, compression type, and audio duration of the audio data. The parameters are passed in with this call.
- setProperties( ): reads the audio data to get the values of the object attributes and then stores them in the object. For the known attributes that ORDAudio understands, it sets the properties for these attributes. These include: format, encoding type, data type, number of channels, sampling rate, and sample size of the audio data.
- checkProperties( ): calls the format plug-in to check the properties including format, encoding type, number of channels, sampling rate, and sample size of the audio data, and returns a Boolean value TRUE if the properties stored in object attributes match those in the audio data.
- getAttribute( ): returns the value of the requested attribute. This method is only available for user-defined format plug-ins.
- getAllAttributes( ): returns a formatted string for convenient client access. For natively supported formats, the string includes the following list of audio data attributes separated by a comma (','): FileFormat, Encoding, NumberOfChannels, SamplingRate, and SampleSize. Different format plug-ins can return data in any format in this call. For user-defined formats, the string is defined by the format plug-in.

ORDAudio Methods Associated with Processing Audio Data

- processAudioCommand( ): sends commands and related arguments to the format plug-in for processing. This method is available only for user-defined format plug-ins.

For more information on object types and methods, see the *Oracle8i Concepts* manual.

3.2.1 Example Table Definitions

The methods described in this reference chapter show examples based on a test audio table TAUD. Refer to the TAUD table definition that follows when reading through the examples in Section 3.2.2 through Section 3.2.9:

TAUD Table Definition

```
CREATE TABLE TAUD(n NUMBER, aud ORDSYS.ORDAUDIO)
storage (initial 100K next 100K pctincrease 0);

INSERT INTO TAUD VALUES(1, ORDSYS.ORDAudio(
                           NULL,
        ORDSYS.ORDSOURCE(EMPTY_BLOB(), NULL, NULL, NULL, SYSDATE, NULL),
                           NULL,
                           NULL,
                           EMPTY_CLOB(), NULL, 0, 0, 0, NULL, 0)
                       );
INSERT INTO TAUD VALUES(2, ORDSYS.ORDAudio(
                           NULL,
        ORDSYS.ORDSOURCE(EMPTY_BLOB(), NULL, NULL, NULL, SYSDATE, NULL),
                           NULL,
                           NULL,
                           EMPTY_CLOB(), NULL, 0, 0, 0, NULL, 0)
                       );

CREATE TABLE TS(n NUMBER, s ORDSYS.ORDSOURCE);
INSERT INTO TS VALUES(1, ORDSYS.ORDSOURCE(EMPTY_BLOB(), NULL, NULL, NULL, SYSDATE, NULL));
INSERT INTO TS VALUES(2, ORDSYS.ORDSOURCE(EMPTY_BLOB(), NULL, NULL, NULL, SYSDATE, NULL));
INSERT INTO TS VALUES(3, ORDSYS.ORDSOURCE(EMPTY_BLOB(), NULL, NULL, NULL, SYSDATE, NULL));
INSERT INTO TS VALUES(4, ORDSYS.ORDSOURCE(EMPTY_BLOB(), NULL, NULL, NULL, SYSDATE, NULL));
```

3.2.2 ORDAudio Methods Associated with the updateTime Attribute

This section presents reference information on the ORDAudio methods associated with the updateTime attribute.

getUpdateTime Method

Format

```
getUpdateTime RETURN DATE;
```

Description

Returns the time when the object was last updated.

Parameters

none

Usage

none

Pragmas

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

Exception

none

Example

Get the updated time of some audio data:

```
DECLARE
 obj ORDSYS.ORDAudio;
BEGIN
 SELECT aud INTO obj FROM TAUD WHERE N = 1 ;
 DBMS_OUTPUT.PUT_LINE(TO_CHAR(obj.getUpdateTime,'MM-DD-YYYY HH24:MI:SS'));
  EXCEPTION
        WHEN ORDSYS.ORDAudioExceptions.METHOD_NOT_SUPPORTED THEN
        DBMS_OUTPUT.PUT_LINE('Source not specified');
END;
/
```

setUpdateTime( ) Method

Format

```
setUpdateTime(current_time DATE);
```

Description

Sets the time when the audio data was last updated. Use this method whenever you modify the audio data. The methods setDescription( ), setMimeType( ), setSource( ), import( ), importFrom( ), export( ), deleteContent, and all set audio accessors call this method implicitly.

Parameters

current_time
The timestamp to be stored. Defaults to SYSDATE.

Usage

You must invoke this method whenever you modify the audio data.

Pragmas

none

Exception

none

Example

Set the updated time of some audio data:

```
DECLARE
 obj ORDSYS.ORDAudio;
BEGIN
 SELECT aud INTO obj FROM TAUD WHERE N = 1;
 obj.setUpdateTime(SYSDATE);
 UPDATE TAUD SET aud=obj WHERE N = 1;
 COMMIT;
END;
/
```

3.2.3 ORDAudio Methods Associated with the description Attribute

This section presents reference information on the ORDAudio methods associated with the description attribute.

setDescription( ) Method

Format

```
setDescription (user_description IN VARCHAR2);
```

Description

Sets the description of the audio data.

Parameters

user_description
The description of the audio data.

Usage

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 Oracle8*i* *inter*-Media make use of this description attribute to present audio data to users.

Calling this method implicitly calls the setUpdateTime( ) method.

Pragmas

none

Exception

none

Example

Set the description attribute for some audio data:

```
DECLARE
  obj ORDSYS.ORDAudio;
BEGIN
  SELECT aud INTO obj FROM TAUD  WHERE N=1 FOR UPDATE;
  DBMS_OUTPUT.PUT_LINE('writing title');
  DBMS_OUTPUT.PUT_LINE('-------------');
```

```
  obj.setDescription('audio1.wav');
  DBMS_OUTPUT.PUT_LINE(obj.getDescription);
  UPDATE TAUD SET aud=obj WHERE N=1;
  COMMIT;
END;
/
```

getDescription Method

Format

```
getDescription RETURN VARCHAR2;
```

Description

Returns the description of the audio data.

Parameters

none

Usage

none

Pragmas

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

Exception

If the description is not set and you call the getDescription( ) method, a DESCRIPTION_IS_NOT_SET exception is raised.

Example

See the example in the setDescription( ) Method on page 3-17.

3.2.4 ORDAudio Methods Associated with the mimeType Attribute

This section presents reference information on the ORDAudio methods associated with the mimeType attribute.

setMimeType( ) Method

Format

```
setMimeType(mime IN VARCHAR2);
```

Description

Allows you to set the MIME type of the audio data.

Parameters

mime
The MIME type.

Usage

Calling this method implicitly calls the setUpdateTime( ) method.

Call the setMimeType( ) method to set the MIME type if the source is a file or BLOB.

The MIME type is extracted from the HTTP header on import for HTTP sources.

Pragmas

none

Exception

If you call the setMimeType( ) method and the value for mimeType is NULL, an INVALID_MIME_TYPE exception is raised.

Example

Set the MIME type for some stored audio data:

```
DECLARE
  obj ORDSYS.ORDAudio;
BEGIN
  SELECT aud INTO obj FROM TAUD  WHERE N=1 FOR UPDATE;
  DBMS_OUTPUT.PUT_LINE('writing mimetype');
  DBMS_OUTPUT.PUT_LINE('----------------');
  obj.setMimeType('audio/basic');
  DBMS_OUTPUT.PUT_LINE(obj.getMimeType);
```

```
  UPDATE TAUD SET aud=obj WHERE N=1;
  COMMIT;
END;
/
```

getMimeType Method

Format

```
getMimeType RETURN VARCHAR2;
```

Description

Returns the MIME type for the audio data.

Parameters

none

Usage

If the source is an HTTP server, the MIME type information is read from the HTTP header information. If the source is a file or BLOB, you must call the setMimeType( ) method and set the MIME type.

Pragmas

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

Exception

none

Example

See the example in the setMimeType( ) Method on page 3-21.

3.2.5 ORDAudio Methods Associated with the source Attribute

This section presents reference information on the ORDAudio methods associated with the source attribute.

processSourceCommand( ) Method

Format

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

Description

Allows you to 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 must be allocated. You must call the openSource( ) method; see the introduction to this chapter for more information.

cmd
Any command recognized by the source plug-in.

arguments
The arguments of the command.

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

Usage

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

Exception

If the value of srcType is NULL, then calling this method raises the exception ORDSourceExceptions.INCOMPLETE_SOURCE_INFORMATION.

If the processSourceCommand( ) method is not supported by the source plug-in being used, an ORDSourceExceptions.METHOD_NOT_SUPPORTED exception is raised.

Calling this method within a source plug-in when any other exception is raised, raises an exception, ORDSourceExceptions.SOURCE_PLUGIN_EXCEPTION.

See Appendix H for more information about these exceptions.

Example

Process some commands:

```
DECLARE
  obj ORDSYS.ORDAudio;
  res RAW(4000);
  result RAW(4000);
  command VARCHAR(4000);
  argList VARCHAR(4000);
  ctx RAW(4000) :=NULL;
BEGIN
select aud into obj from TAUD where N =1 for UPDATE;
-- assign command
-- assign argList
res := obj.processSourceCommand (ctx, command, argList, result);
UPDATE TAUD SET aud=obj WHERE N=1 ;
COMMIT;
  EXCEPTION
        WHEN ORDSYS.ORDSourceExceptions.METHOD_NOT_SUPPORTED THEN
         DBMS_OUTPUT.put_line('Source METHOD_NOT_SUPPORTED caught');
        WHEN ORDSYS.ORDSourceExceptions.SOURCE_PLUGIN_EXCEPTION THEN
         DBMS_OUTPUT.put_line('SOURCE PLUGIN EXCEPTION caught');
        WHEN ORDSYS.ORDAudioExceptions.METHOD_NOT_SUPPORTED THEN
         DBMS_OUTPUT.put_line('AUDIO METHOD_NOT_SUPPORTED EXCEPTION caught');
        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;
/
```

isLocal Method

Format

```
isLocal RETURN BOOLEAN;
```

Description

Returns TRUE if the data is stored locally in a BLOB or FALSE if the data is stored externally.

Parameters

none

Usage

If the local attribute is set to 1 or NULL, this method returns TRUE, otherwise this method returns FALSE.

Pragmas

Pragma RESTRICT_REFERENCES(getLocal, WNDS, WNPS, RNDS, RNPS)

Exception

none

Example

Determine whether or not the data is local:

```
DECLARE
 obj ORDSYS.ORDAudio;
BEGIN
 SELECT s INTO obj FROM TAUD WHERE N = 1 ;
 if(obj.isLocal = TRUE)  then
    DBMS_OUTPUT.put_line('local is set true');
 else
    DBMS_OUTPUT.put_line('local is set false');
 end if;
END;
/
```

setLocal Method

Format

```
setLocal;
```

Description

Sets the local attribute to indicate that the data is stored internally in a BLOB. When local is set, audio methods look for audio data in the source.localData attribute.

Parameters

none

Usage

This method sets the local attribute to 1 meaning the data is stored locally in local-Data.

Pragmas

none

Exception

none

Example

Set the flag to local for the data:

```
DECLARE
 obj ORDSYS.ORDAudio;
BEGIN
 SELECT s INTO obj FROM TAUD WHERE N = 1 FOR UPDATE;
 obj.setLocal;
 UPDATE TAUD SET s=obj WHERE N = 1;
 COMMIT;
END;
/
```

clearLocal Method

Format

```
clearLocal;
```

Description

Resets the local flag to indicate that the data is stored externally. When the local flag is set to clear, audio methods look for audio data using the srcLocation, srcName, and srcType attributes.

Parameters

none

Usage

This method sets the local attribute to a 0, meaning the data is stored externally or outside of Oracle8*i*.

Pragmas

none

Exception

none

Example

Clear the value of the local flag for the data:

```
DECLARE
 obj ORDSYS.ORDAudio;
BEGIN
 SELECT s INTO obj FROM TAUD WHERE N = 1 FOR UPDATE;
 obj.clearLocal;
 UPDATE TAUD SET s=obj WHERE N = 1;
 COMMIT;
END;
/
```

setSource( ) Method

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 audio data.

Parameters

source_type
The source type of the external data. See the "ORDSource Object Type" definition in Chapter 6 for more information.

source_location
The source location of the external data. See the "ORDSource Object Type" definition in Chapter 6 for more information.

source_name
The source name of the external data. See the "ORDSource Object Type" definition in Chapter 6 for more information.

Usage

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

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

Calling this method implicitly calls the setUpdateTime( ) method.

Pragmas

none

Exception

none

Example

Set the source of the audio data:

```
DECLARE
  obj ORDSYS.ORDAudio;
BEGIN
  SELECT aud INTO obj FROM TAUD  WHERE N=1 FOR UPDATE;
  DBMS_OUTPUT.PUT_LINE('setting and getting source');
  DBMS_OUTPUT.PUT_LINE('-------------------------');
  obj.setSource('FILE','AUDIODIR','audio.au');
  DBMS_OUTPUT.PUT_LINE(obj.getSource);
  UPDATE TAUD SET aud=obj WHERE N=1;
  COMMIT;
END;
/
```

getSource Method

Format

```
getSource RETURN VARCHAR2;
```

Description

Returns information about the external location of the audio data in URL format.

Parameters

none

Usage

Possible return values are:

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

Pragmas

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

Exception

none

Example

See the example in the setSource( ) Method on page 3-31.

getSourceType Method

Format

```
getSourceType RETURN VARCHAR2;
```

Description

Returns a string containing the type of the external audio data source.

Parameters

none

Usage

Returns a VARCHAR2 string containing the type of the external audio data source, for example 'FILE'.

Pragmas

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

Exception

none

Example

Get the source type information about an audio data source:

```
DECLARE
  obj ORDSYS.ORDAudio;
BEGIN
  SELECT aud INTO obj FROM TAUD  WHERE N=1 FOR UPDATE;
  DBMS_OUTPUT.PUT_LINE('setting and getting source');
  DBMS_OUTPUT.PUT_LINE('-------------------------');
  -- set source to a file
  obj.setSource('FILE','AUDIODIR','testaud.dat');
  -- get source information
  DBMS_OUTPUT.put_line(obj.getSource);
  DBMS_OUTPUT.put_line(obj.getSourceType);
  DBMS_OUTPUT.put_line(obj.getSourceLocation);
  DBMS_OUTPUT.put_line(obj.getSourceName);
```

```
  UPDATE TAUD SET aud=obj WHERE N=1;
  COMMIT;
  EXCEPTION
        WHEN ORDSYS.ORDAudioExceptions.METHOD_NOT_SUPPORTED THEN
        DBMS_OUTPUT.PUT_LINE('Source not specified');
END;
/
```

getSourceLocation Method

Format

```
getSourceLocation RETURN VARCHAR2;
```

Description

Returns a string containing the value of the external audio data source location.

Parameters

none

Usage

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

Pragmas

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

Exception

If the value of srcLocation is NULL, then calling this method raises an ORDSource-Exceptions.INCOMPLETE_SOURCE_LOCATION exception.

Example

See the example in the getSourceType Method on page 3-33.

getSourceName Method

Format

```
getSourceName RETURN VARCHAR2;
```

Description

Returns a string containing of the name of the audio external data source.

Parameters

none

Usage

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

Pragmas

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

Exception

If the value of srcName is NULL, then calling this method raises an ORDSourceExceptions.INCOMPLETE_SOURCE_NAME exception.

Example

See the example in the getSourceType Method on page 3-33.

import( ) Method

Format

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

Description

Transfers audio data from an external audio data source to a local source (local-Data) within an Oracle database.

Parameters

ctx
The source plug-in context information.This must be allocated. You must call the openSource( ) method; see the introduction to this chapter for more information.

Usage

Use the setSource( ) method to set the external source type, location, and name prior to calling the import( ) method.

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

After importing data from an external audio data source to a local source (within an 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.

Pragmas

none

Exception

If the value of srcType is NULL, then calling this method raises an ORDSourceExceptions.INCOMPLETE_SOURCE_INFORMATION exception.

If the value of dlob is NULL, then calling this method raises an ORDSourceExceptions.NULL_SOURCE exception.

If the import( ) method is not supported by the source plug-in being used, an ORDSourceExceptionsMETHOD_NOT_SUPPORTED exception is raised.

Calling this method within a source plug-in when any other exception is raised, raises an ORDSourceExceptions.SOURCE_PLUGIN_EXCEPTION exception.

See Appendix H for more information about these exceptions.

Example

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

```
DECLARE
  obj ORDSYS.ORDAudio;
  ctx RAW(4000) :=NULL;
BEGIN
  SELECT aud INTO obj FROM TAUD  WHERE N=1 FOR UPDATE;
  DBMS_OUTPUT.PUT_LINE('setting and getting source');
  DBMS_OUTPUT.PUT_LINE('-------------------------');
  -- set source to a file
  obj.setSource('FILE','AUDIODIR','testaud.dat');
  -- get source information
  DBMS_OUTPUT.PUT_LINE(obj.getSource);
  -- import data
  obj.import(ctx);
  -- check size
  DBMS_OUTPUT.PUT_LINE(TO_CHAR(obj.getContentLength(ctx)));
  DBMS_OUTPUT.PUT_LINE(obj.getSource);
  DBMS_OUTPUT.PUT_LINE('deleting contents');
  DBMS_OUTPUT.PUT_LINE('----------------');
  obj.deleteContent;
  DBMS_OUTPUT.PUT_LINE(TO_CHAR(obj.getContentLength(ctx)));
  UPDATE TAUD SET aud=obj WHERE N=1;
  COMMIT;
END;
/
```

importFrom( ) Method

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 a local source (localData) within an Oracle database.

Parameters

ctx
The source plug-in context information. This must be allocated. You must call the openSource( ) method; see the introduction to this chapter for more information.

source_type
The source type of the audio data.

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

source_name
The name of the audio data.

Usage

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

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

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

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

Pragmas

none

Exception

If the value of dlob is NULL, then calling this method raises an ORDSourceExceptions.NULL_SOURCE exception.

If the importFrom( ) method is not supported by the source plug-in being used, an ORDSourceExceptionsMETHOD_NOT_SUPPORTED exception is raised.

Calling this method within a source plug-in when any other exception is raised, raises an ORDSourceExceptions.SOURCE_PLUGIN_EXCEPTION exception.

See Appendix H for more information about these exceptions.

Example

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

```
DECLARE
  obj ORDSYS.ORDAudio;
  ctx RAW(4000) :=NULL;
BEGIN
  SELECT aud INTO obj FROM TAUD  WHERE N=1 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','AUDIODIR','testaud.dat');
  -- check size
  DBMS_OUTPUT.PUT_LINE(TO_CHAR(obj.getContentLength(ctx)));
  DBMS_OUTPUT.PUT_LINE(obj.getSource);
  DBMS_OUTPUT.PUT_LINE('deleting contents');
  DBMS_OUTPUT.PUT_LINE('----------------');
  obj.deleteContent;
  DBMS_OUTPUT.PUT_LINE(TO_CHAR(obj.getContentLength(ctx)));
  DBMS_OUTPUT.PUT_LINE(TO_CHAR(DBMS_LOB.GETLENGTH(obj.getContent)));
  UPDATE TAUD SET aud=obj WHERE N=1;
  COMMIT;
  EXCEPTION
        WHEN ORDSYS.ORDAudioExceptions.METHOD_NOT_SUPPORTED THEN
        DBMS_OUTPUT.PUT_LINE('Source not specified');
END;
/
```

export( ) Method

Format

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

Description

Transfers audio data from a local source (localData) within an Oracle Database to an external audio data source.

Parameters

ctx
The source plug-in context information.

source_type
The source type of the location to where the data is to be exported.

source_location
The location where the audio data is to be exported.

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

Usage

After exporting audio data, all audio attributes remain unchanged and srcType, srcLocation, and srcName are updated with input values. After calling the export( ) method, you can call the deleteContent method to delete the content of the local data.

There is no server-side, native support for the export( ) method; this method is available for user-defined sources that can support the export( ) method.

Invoking this method implicitly calls the setUpdateTime( ) method.

Pragmas

none

Exception

If the value of srcType is NULL, then calling this method raises an ORDSourceExceptions.INCOMPLETE_SOURCE_INFORMATION exception.

If the export( ) method is not supported by the source plug-in being used, an ORDSourceExceptions.METHOD_NOT_SUPPORTED exception is raised.

Calling this method within a source plug-in when any other exception is raised, raises an ORDSourceExceptions.SOURCE_PLUGIN_EXCEPTION exception.

See Appendix H for more information about these exceptions.

Example

Export audio data from a local source to an external audio data source and then delete the contents:

```
DECLARE
  obj ORDSYS.ORDAudio;
  ctx RAW(4000) :=NULL;
BEGIN
   SELECT aud INTO obj FROM TAUD   WHERE N =1;
   obj.export(ctx,'FILE','AUDIODIR','complete.wav');
  EXCEPTION
  WHEN ORDSYS.ORDSourceExceptions.METHOD_NOT_SUPPORTED THEN
   DBMS_OUTPUT.put_line('Source METHOD_NOT_SUPPORTED caught');
  WHEN ORDSYS.ORDSourceExceptions.SOURCE_PLUGIN_EXCEPTION THEN
   DBMS_OUTPUT.put_line('SOURCE PLUGIN EXCEPTION caught');
  WHEN ORDSYS.ORDAudioExceptions.METHOD_NOT_SUPPORTED THEN
   DBMS_OUTPUT.put_line('AUDIO METHOD_NOT_SUPPORTED EXCEPTION caught');
  WHEN ORDSYS.ORDAudioExceptions.AUDIO_PLUGIN_EXCEPTION THEN
   DBMS_OUTPUT.put_line('AUDIO PLUGIN EXCEPTION caught');
  WHEN OTHERS THEN
   DBMS_OUTPUT.put_line('OTHER EXCEPTION caught');
END;
/
```

getContentLength( ) Method

Format

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

Description

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

Parameters

ctx
The source plug-in context information.

Usage

This method is not supported for all source types. For example, "HTTP" type sources do not support this method. If you want to implement this call for "HTTP" type sources, you must define your own modified "HTTP" source type and implement this method on it.

Pragmas

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

Exception

If the value of srcType is NULL and data is not stored locally in the BLOB, then calling this method raises an exception, ORDSourceExceptions.INCOMPLETE_SOURCE_INFORMATION.

Calling this method within a source plug-in when any other exception is raised, raises an ORDSourceExceptions.SOURCE_PLUGIN_EXCEPTION exception.

See Appendix H for more information about this exception.

Example

See the example in the import( ) Method on page 3-38

getContentInLob( ) Method

Format

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

Description

Transfers data from a data source into the specified BLOB. The BLOB can be another BLOB, not the one for the object.

Parameters

ctx
The source plug-in context information.

dest_lob
The LOB in which to receive data.

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

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

Usage

none

Pragmas

none

Exception

If the value of srcType is NULL, then calling this method raises an ORDSourceExceptions.INCOMPLETE_SOURCE_INFORMATION exception.

If the getContentInLob( ) method is not supported by the source plug-in being used, an ORDSourceExceptions.METHOD_NOT_SUPPORTED exception is raised.

Calling this method within a source plug-in when any other exception is raised, raises an ORDSourceExceptions.SOURCE_PLUGIN_EXCEPTION exception.

See Appendix H for more information about these exceptions.

Example

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

```
DECLARE
 obj ORDSYS.ORDAudio;
 tempBLob BLOB;
 mimeType VARCHAR2(4000);
 format VARCHAR2(4000);
 ctx RAW(4000) :=NULL;
BEGIN
 SELECT aud INTO obj FROM TAUD WHERE N = 1 ;
 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);
 -- process tempBLob
 DBMS_OUTPUT.PUT_LINE(TO_CHAR(DBMS_LOB.getLength(tempBLob)));
 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;
/
```

getContent Method

Format

```
getContent RETURN BLOB;
```

Description

Returns a handle to the local BLOB storage, that is the BLOB within the ORDAudio object.

Parameters

none

Usage

none

Pragmas

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

Exception

none

Example

A client accesses audio data:

```
DECLARE
  obj ORDSYS.ORDAudio;
BEGIN
  SELECT aud INTO obj FROM TAUD  WHERE N=1 ;
  DBMS_OUTPUT.PUT_LINE('-------------------------');
  DBMS_OUTPUT.put_line(TO_CHAR(DBMS_LOB.getLength(obj.getContent)));
  EXCEPTION
        WHEN ORDSYS.ORDAudioExceptions.METHOD_NOT_SUPPORTED THEN
        DBMS_OUTPUT.PUT_LINE('Source not specified');
END;
/
```

deleteContent Method

Format

```
deleteContent;
```

Description

Deletes the local data from the current local source (localData).

Parameters

none

Usage

This method can be called after you export the data from the local source to an external audio 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

Exception

none

Example

See the example in the import( ) Method on page 3-38.

3.2.6 ORDAudio Methods Associated with File-Like Operations

This section presents reference information on the ORDAudio methods associated with file-like operations on a data source. You can use the following methods specifically to manipulate audio data.

openSource( ) Method

Format

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

Description

Opens a data source.

Parameters

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

ctx
The source plug-in context information. This must be allocated. You must call the openSource( ) method; see the introduction to this chapter for more information.

Usage

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

Pragmas

none

Exception

If the value for srcType is NULL and data is not local, then calling this method raises an ORDSourceExceptions.INCOMPLETE_SOURCE_INFORMATION exception.

If the openSource( ) method is not supported by the source plug-in being used, an ORDSourceExceptions.METHOD_NOT_SUPPORTED exception is raised.

Calling this method within a source plug-in when any other exception is raised, raises an ORDSourceExceptions.SOURCE_PLUGIN_EXCEPTION exception.

See Appendix H for more information about these exceptions.

Example

Open an external data source:

```
DECLARE
  obj ORDSYS.ORDAudio;
  res INTEGER;
  ctx RAW(4000) :=NULL;
  userArg RAW(4000);
BEGIN
  select aud into obj from TAUD where N =1 for UPDATE;
  res := obj.openSource(userArg, ctx);
  UPDATE TAUD SET aud=obj WHERE N=1 ;
  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;
/
```

closeSource( ) Method

Format

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

Description

Closes a data source.

Parameters

ctx
The source plug-in context information. You must call the openSource( ) method; see the introduction to this chapter for more information.

Usage

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

Pragmas

none

Exception

If the value for srcType is NULL and data is not local, then calling this method raises an ORDSourceExceptions.INCOMPLETE_SOURCE_INFORMATION exception.

If the closeSource( ) method is not supported by the source plug-in being used, an ORDSourceExceptions.METHOD_NOT_SUPPORTED exception is raised.

Calling this method within a source plug-in when any other exception is raised, raises an ORDSourceExceptions.SOURCE_PLUGIN_EXCEPTION exception.

See Appendix H for more information about these exceptions.

Example

Close an external data source:

```
DECLARE
  obj ORDSYS.ORDAudio;
  res INTEGER;
  ctx RAW(4000) :=NULL;
BEGIN
  select aud into obj from TAUD where N =2 for UPDATE;
  res := obj.closeSource(ctx);
  UPDATE TAUD SET aud=obj WHERE N=2 ;
  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;
/
```

trimSource( ) Method

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 must be allocated. You must call the openSource( ) method; see the introduction to this chapter for more information.

newlen
The trimmed new length.

Usage

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

Pragmas

none

Exception

If the value for srcType is NULL and data is not local, then calling this method raises an ORDSourceExceptions.INCOMPLETE_SOURCE_INFORMATION exception.

If the trimSource( ) method is not supported by the source plug-in being used, an ORDSourceExceptions.METHOD_NOT_SUPPORTED exception is raised.

Calling this method within a source plug-in when any other exception is raised, raises an ORDSourceExceptions.SOURCE_PLUGIN_EXCEPTION exception.

See Appendix H for more information about these exceptions.

Example

Trim an external data source:

```
DECLARE
  obj ORDSYS.ORDAudio;
  res INTEGER;
  ctx RAW(4000) :=NULL;
BEGIN
  select aud into obj from TAUD where N =1 for UPDATE;
  res := obj.trimSource(ctx,0);
  UPDATE TAUD SET aud=obj WHERE N=1 ;
  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;
/
```

readFromSource( ) Method

Format

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

Description

Allows you to read a buffer of n bytes from a source beginning at a start position.

Parameters

ctx
The source plug-in context information. This must be allocated. You must call the openSource( ) method; see the introduction to this chapter for more information.

startPos
The start position in the data source.

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

buffer
The buffer into which the data will be read.

Usage

This method is not supported for HTTP sources.

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

Pragmas

none

Exception

If the value of srcType is NULL and data is not local, then calling this method raises an ORDSourceExceptions.INCOMPLETE_SOURCE_INFORMATION exception.

If data is local but localData is NULL, then calling this method raises an ORD-SourceExceptions.NULL_SOURCE exception.

If the readFromSource( ) method is not supported by the source plug-in being used, an ORDSourceExceptions.METHOD_NOT_SUPPORTED exception is raised.

Calling this method within a source plug-in when any other exception is raised, raises an ORDSourceExceptions.SOURCE_PLUGIN_EXCEPTION exception.

See Appendix H for more information about these exceptions.

Example

Read a buffer from the source:

```
DECLARE
  obj ORDSYS.ORDAudio;
  buffer RAW(4000);
  i INTEGER;
  ctx RAW(4000) :=NULL;
BEGIN
  i := 20;
  select aud into obj from TAUD where N =1 ;
  obj.readFromSource(ctx,1,i,buffer);
  DBMS_OUTPUT.PUT_LINE(TO_CHAR(obj.getContentLength(ctx)));
  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;
/
```

writeToSource( ) Method

Format

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

Description

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

Parameters

ctx
The source plug-in context information. This must be allocated. You must call the openSource( ) method; see the introduction to this chapter for more information.

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

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

buffer
The buffer of data to be written.

Usage

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

Pragmas

none

Exception

If the value of srcType is NULL and data is not local, then calling this method raises an ORDSourceExceptions.INCOMPLETE_SOURCE_INFORMATION exception.

If data is local but localData is NULL, then calling this method raises an ORD-SourceExceptions.NULL_SOURCE exception.

If the writeToSource( ) method is not supported by the source plug-in being used, an ORDSourceExceptions.METHOD_NOT_SUPPORTED exception is raised.

Calling this method within a source plug-in when any other exception is raised, raises an ORDSourceExceptions.SOURCE_PLUGIN_EXCEPTION exception.

See Appendix H for more information about these exceptions.

Example

Write a buffer to the source:

```
DECLARE
  obj ORDSYS.ORDAudio;
  n INTEGER := 6;
  ctx RAW(4000) :=NULL;
BEGIN
  select aud into obj from TAUD where N =1 for update;
  obj.writeToSource(ctx,1,n,UTL_RAW.CAST_TO_RAW('helloP'));
  DBMS_OUTPUT.PUT_LINE(TO_CHAR(obj.getContentLength(ctx)));
  update  TAUD set aud = obj where  N =1 ;
  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;
/
```

3.2.7 ORDAudio Methods Associated with the comments Attribute

This section presents reference information on the ORDAudio methods associated with the comments attribute.

appendToComments( ) Method

Format

```
appendToComments(amount IN BINARY_INTEGER,
                 buffer IN VARCHAR2);
```

Description

Appends a specified buffer and amount of comment data to the end of the comments attribute of the audio object.

Parameters

amount
The amount of comment data to be appended.

buffer
The buffer of comment data to be appended.

Usage

none

Pragmas

none

Exception

The exceptions-raised behavior for this method is similar to that of the DBMS_LOB functions and procedures. See the DBMS_LOB package description in the *Oracle8i Application Developer's Reference - Packages* manual for a list of exceptions that can be raised for the DBMS_LOB functions and procedures.

Example

Append comment information to the comments attribute of the audio object:

```
DECLARE
  obj ORDSYS.ORDAudio;
  i INTEGER;
  j INTEGER;
```

```
BEGIN
  SELECT aud INTO obj FROM TAUD  WHERE N=1 FOR UPDATE;
  obj.deleteComments;
  obj.writeToComments(1,18,'This is a Comments');
  obj.appendToComments(18,'This is a Comments');
  -- check comments
  DBMS_OUTPUT.PUT_LINE(obj.readFromComments(1,obj.getCommentLength));
  DBMS_OUTPUT.PUT_LINE(obj.locateInComments('Comments',1));
  obj.trimComments(18);
  DBMS_OUTPUT.PUT_LINE(obj.readFromComments(1,18));
  i := 8;
  j := 9;
  obj.eraseFromComments(i,j);
  DBMS_OUTPUT.PUT_LINE(obj.readFromComments(1,10));
  obj.deleteComments;
  DBMS_OUTPUT.PUT_LINE(obj.readFromComments(1,10));
  UPDATE TAUD SET aud=obj WHERE N=1;
  COMMIT;
  EXCEPTION
        WHEN ORDSYS.ORDAudioExceptions.METHOD_NOT_SUPPORTED THEN
        DBMS_OUTPUT.PUT_LINE('Source not specified');
END;
/
```

writeToComments( ) Method

Format

```
writeToComments(offset IN INTEGER,
                amount IN BINARY_INTEGER,
                buffer IN VARCHAR2);
```

Description

Writes a specified amount of comment buffer data to the comments attribute of the audio object beginning at the specified offset.

Parameters

offset
The starting offset position in comments where comments data is to be written.

amount
The amount of comment data to be written.

buffer
The buffer of comment data to be written.

Usage

none

Pragmas

none

Exception

The exceptions-raised behavior for this method is similar to that of the DBMS_LOB functions and procedures. See the DBMS_LOB package description in the *Oracle8i Application Developer's Reference - Packages* manual for a list of exceptions that can be raised for the DBMS_LOB functions and procedures.

Example

See the example in the appendToComments( ) Method on page 3-60.

readFromComments( ) Method

Format

```
readFromComments(offset IN INTEGER,
                 amount IN BINARY_INTEGER :=32767)
RETURN VARCHAR2;
```

Description

Reads a specified amount of comment data from the comments attribute of the audio object beginning at a specified offset.

Parameters

offset
The starting offset position in comments from where comments data is to be read.

amount
The amount of comment data to be read.

Usage

none

Pragmas

Pragma RESTRICT_REFERENCES(readFromComments, WNDS, WNPS, RNDS, RNPS)

Exception

The exceptions-raised behavior for this method is similar to that of the DBMS_LOB functions and procedures. See the DBMS_LOB package description in the *Oracle8i Application Developer's Reference - Packages* manual for a list of exceptions that can be raised for the DBMS_LOB functions and procedures.

Example

See the example in the appendToComments( ) Method on page 3-60.

locateInComments( ) Method

Format

```
locateInComments(pattern    IN VARCHAR2,
                 offset     IN INTEGER := 1,
                 occurrence IN INTEGER := 1)
RETURN INTEGER;
```

Description

Matches and locates the nth occurrence of the specified pattern of character data in the comments attribute of the audio object beginning at a specified offset.

Parameters

pattern
The pattern of comment data for which to search.

offset
The starting offset position in comments where the search for a match should begin.

occurrence
The nth occurrence in the comments where the pattern of comment data was found.

Usage

none

Pragmas

none

Exception

The exceptions-raised behavior for this method is similar to that of the DBMS_LOB functions and procedures. See the DBMS_LOB package description in the *Oracle8i Application Developer's Reference - Packages* manual for a list of exceptions that can be raised for the DBMS_LOB functions and procedures.

Example

See the example in the appendToComments( ) Method on page 3-60.

trimComments( ) Method

Format

```
trimComments(newlen IN INTEGER);
```

Description

Trims the length of comments of the audio object to the specified new length.

Parameters

newlen
The new length to which the comments are to be trimmed.

Usage

none

Pragmas

none

Exception

The exceptions-raised behavior for this method is similar to that of the DBMS_LOB functions and procedures. See the DBMS_LOB package description in the *Oracle8i Application Developer's Reference - Packages* manual for a list of exceptions that can be raised for the DBMS_LOB functions and procedures.

Example

See the example in the appendToComments( ) Method on page 3-60.

eraseFromComments( ) Method

Format

```
eraseFromComments(amount IN OUT NOCOPY INTEGER,
                  offset IN INTEGER := 1);
```

Description

Erases a specified amount of comment data from the comments attribute of the audio object beginning at a specified offset.

Parameters

amount
The amount of comment data to be erased.

offset
The starting offset position in comments where comments data is to be erased.

Usage

none

Pragmas

none

Exception

The exceptions-raised behavior for this method is similar to that of the DBMS_LOB functions and procedures. See the DBMS_LOB package description in the *Oracle8i Application Developer's Reference - Packages* manual for a list of exceptions that can be raised for the DBMS_LOB functions and procedures.

Example

See the example in the appendToComments( ) Method on page 3-60.

deleteComments Method

Format

```
deleteComments;
```

Description

Deletes the comments attribute of the audio object.

Parameters

none

Usage

none

Pragmas

none

Exception

The exceptions-raised behavior for this method is similar to that of the DBMS_LOB functions and procedures. See the DBMS_LOB package description in the *Oracle8i Application Developer's Reference - Packages* manual for a list of exceptions that can be raised for the DBMS_LOB functions and procedures.

Example

See the example in the appendToComments( ) Method on page 3-60.

loadCommentsFromFile( ) Method

Format

```
loadCommentsFromFile(fileobj  IN BFILE,
                     amount   IN INTEGER,
                     from_loc IN INTEGER := 1,
                     to_loc   IN INTEGER := 1);
```

Description

Loads a specified amount of comment data from a BFILE into the comments attribute of the audio object beginning at a specified offset.

Parameters

fileobj
The file object to be loaded.

amount
The amount of comment data to be loaded from the BFILE.

from_loc
The location from which to load comments from the BFILE.

to_loc
The location to which to load comments.

Usage

none

Pragmas

none

Exception

The exceptions-raised behavior for this method is similar to that of the DBMS_LOB functions and procedures. See the DBMS_LOB package description in the *Oracle8i Application Developer's Reference - Packages* manual for a list of exceptions that can be raised for the DBMS_LOB functions and procedures.

Example

Load comment information from a BFILE into the comments of the audio data:

```
DECLARE
  file_handle BFILE;
  obj ORDSYS.ORDAudio;
  isopen BINARY_INTEGER;
  amount INTEGER;
BEGIN
  SELECT aud INTO obj FROM TAUD  WHERE N=1 FOR UPDATE;
  file_handle := BFILENAME(obj.getSourceLocation, obj.getSourceName);
     isopen := DBMS_LOB.FILEISOPEN(file_handle);
     IF isopen = 0 THEN
       dbms_output.put_line('File Not Open');
       DBMS_LOB.FILEOPEN(file_handle, DBMS_LOB.FILE_READONLY);
     END IF;
     dbms_output.put_line('File is now Open');
     isopen := DBMS_LOB.FILEISOPEN(file_handle);
     IF isopen <> 0 THEN
       amount := DBMS_LOB.GETLENGTH(file_handle);
     END IF;
     obj.deleteComments;
     obj.loadCommentsFromFile(file_handle, 18, 1, 18);
     DBMS_OUTPUT.put_line(TO_CHAR(obj.getCommentLength));
  UPDATE TAUD SET aud=obj WHERE N=1;
  COMMIT;
END;
/
```

copyCommentsOut( ) Method

Format

```
copyCommentsOut(dest     IN OUT NOCOPY CLOB,
                amount   IN INTEGER,
                from_loc IN INTEGER := 1,
                to_loc   IN INTEGER := 1);
```

Description

Copies a specified amount of audio object comments attribute into the given CLOB.

Parameters

dest
The destination to which the comments are to be copied.

amount
The amount of comments to be copied.

from_loc
The location from which to copy comments.

to_loc
The location to which to copy comments.

Usage

none

Pragmas

none

Exception

The exceptions-raised behavior for this method is similar to that of the DBMS_LOB functions and procedures. See the DBMS_LOB package description in the *Oracle8i Application Developer's Reference - Packages* manual for a list of exceptions that can be raised for the DBMS_LOB functions and procedures.

Example

Copy comments of the audio data to the given CLOB:

```
DECLARE
  obj ORDSYS.ORDAudio;
  obj1 ORDSYS.ORDAudio;
BEGIN
  SELECT aud INTO obj1 FROM TAUD  WHERE N=2 FOR UPDATE;
  SELECT aud INTO obj FROM TAUD  WHERE N=1;
  obj.copyCommentsOut(obj1.comments,obj.getCommentLength,1,10);
  DBMS_OUTPUT.put_line(obj1.getCommentLength);
  DBMS_OUTPUT.put_line(obj.getCommentLength);
  UPDATE TAUD SET aud=obj1 WHERE N=2;
  COMMIT;
  EXCEPTION
        WHEN ORDSYS.ORDAudioExceptions.METHOD_NOT_SUPPORTED THEN
        DBMS_OUTPUT.PUT_LINE('Source not specified');
END;
/
```

compareComments( ) Method

Format

```
compareComments(compare_with_lob        IN CLOB,
                amount                  IN INTEGER := 4294967295,
                starting_pos_in_comment IN INTEGER := 1,
                starting_pos_in_compare IN INTEGER := 1)
RETURN INTEGER;
```

Description

Compares a specified amount of comments of audio data with comments of the other CLOB provided.

Parameters

compare_with_lob
The comparison comments.

amount
The amount of comments of audio data to compare with the comparison comments.

starting_pos_in_comment
The starting position in the comments attribute of the audio object.

starting_pos_in_compare
The starting position in the comparison comments.

Usage

none

Pragmas

Pragma RESTRICT_REFERENCES(compareComments, WNDS, WNPS, RNDS, RNPS)

Exception

The exceptions-raised behavior for this method is similar to that of the DBMS_LOB functions and procedures. See the DBMS_LOB package description in the *Oracle8i*

Application Developer's Reference - Packages manual for a list of exceptions that can be raised for the DBMS_LOB functions and procedures.

Example

Compare comments of the audio data with comments of another CLOB:

```
DECLARE
  file_handle BFILE;
  obj ORDSYS.ORDAudio;
  obj1 ORDSYS.ORDAudio;
BEGIN
  SELECT aud INTO obj FROM TAUD  WHERE N=2 ;
  SELECT aud INTO obj1 FROM TAUD  WHERE N=1;
  DBMS_OUTPUT.put_line('comparison output');
  DBMS_OUTPUT.put_line(obj.compareComments(obj1.comments,obj.getCommentLength,1,18));
  EXCEPTION
        WHEN ORDSYS.ORDAudioExceptions.METHOD_NOT_SUPPORTED THEN
        DBMS_OUTPUT.PUT_LINE('Source not specified');
END;
/
```

getCommentLength( ) Method

Format

```
getCommentLength RETURN INTEGER;
```

Description

Returns the length of the comments attribute of the audio object.

Parameters

none

Usage

none

Pragmas

Pragma RESTRICT_REFERENCES(getCommentLength, WNDS, WNPS, RNDS, RNPS)

Exception

The exceptions-raised behavior for this method is similar to that of the DBMS_LOB functions and procedures. See the DBMS_LOB package description in the *Oracle8i Application Developer's Reference - Packages* manual for a list of exceptions that can be raised for the DBMS_LOB functions and procedures.

Example

See the example in the compareComments( ) Method on page 3-73.

3.2.8 ORDAudio Methods Associated with Audio Attributes Accessors

This section presents reference information on the ORDAudio methods associated with the audio attributes accessors.

setFormat( ) Method

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

Calling this method implicitly calls the setUpdateTime( ) method.

Pragmas

none

Exception

If the value for the knownFormat parameter is NULL, then calling this method raises a NULL_INPUT_VALUE exception.

Example

Set the format for some audio data:

```
DECLARE
  obj ORDSYS.ORDAudio;
BEGIN
  select aud into obj from TAUD where N =1 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.METHOD_NOT_SUPPORTED THEN
    DBMS_OUTPUT.put_line('ORDSourceExceptions.METHOD_NOT_SUPPORTED caught');
   WHEN OTHERS THEN
    DBMS_OUTPUT.put_line('EXCEPTION caught');
END;
/
```

getFormat Method

Format

```
getFormat RETURN VARCHAR2;
```

Description

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

Parameters

none

Usage

none

Pragmas

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

Exception

If the value for format is NULL, then calling this method raises an AUDIO_FORMAT_IS_NULL exception.

Example

See the example in the setFormat( ) Method on page 3-76.

getFormat( ) Method

Format

```
getFormat(ctx IN OUT RAW) RETURN VARCHAR2;
```

Description

Calls the format plug-in to read the format embedded in the stored audio data.

Parameters

ctx
The format plug-in context information.

Usage

If the format found in the object is NULL, then the getFormat( ) method uses the default format plug-in to read the audio data to determine the format; otherwise, it uses the plug-in specified by the format. AUFF, AIFF, AIFC, and WAVE plug-ins are provided, so users may use these plug-ins too.

Audio file format information can be extracted from the audio data itself. You can extend support to a file format not known by the ORDAudio object by implementing an ORDPLUGINS.ORDX_<format>_AUDIO package that supports that file format. See Section 2.3.13 for more information.

Pragmas

none

Exception

If the audio plug-in raises an exception when calling this method, then the getFormat( ) method raises an AUDIO_PLUGIN_EXCEPTION exception.

Example

Call the format plug-in to read the actual format embedded in the stored audio data:

```
DECLARE
  obj ORDSYS.ORDAudio;
```

```
  ctx RAW(4000) :=NULL;
BEGIN
  SELECT aud INTO obj FROM TAUD  WHERE N=1;
  DBMS_OUTPUT.PUT_LINE('getting audio file format');
  DBMS_OUTPUT.PUT_LINE('-------------------------');
  DBMS_OUTPUT.PUT_LINE(obj.getFormat(ctx));
  EXCEPTION
  WHEN ORDSYS.ORDAudioExceptions.AUDIO_PLUGIN_EXCEPTION THEN
  DBMS_OUTPUT.PUT_LINE('AUDIO_PLUGIN_EXCEPTION caught');
END;
/
```

setEncoding( ) Method

Format

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

Description

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

Parameters

knownEncoding
A known encoding type.

Usage

Calling this method implicitly calls the setUpdateTime( ) method.

Pragmas

none

Exception

If the value for the knownEncoding parameter is NULL, then calling this method raises a NULL_INPUT_VALUE exception.

Example

See the example in the setFormat( ) Method on page 3-76.

getEncoding Method

Format

```
getEncoding RETURN VARCHAR2;
```

Description

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

Parameters

none

Usage

none

Pragmas

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

Exception

none

Example

See the example in the setFormat( ) Method on page 3-76.

getEncoding( ) Method

Format

```
getEncoding(ctx IN OUT RAW) RETURN VARCHAR2;
```

Description

Calls the format plug-in to read the encoding embedded in the stored audio data.

Parameters

ctx
The format plug-in context information.

Usage

If the format found in the object is NULL, then the getEncoding( ) method uses the default format plug-in to read the audio data to determine the encoding; otherwise, it uses the plug-in specified by the format.

Audio encoding information can be extracted from the audio data itself. You can extend support to a format that is not understood by the ORDAudio object by implementing an ORDPLUGIN.ORDX_<format>_AUDIO package that supports that format. See Section 2.3.13 for more information.

This function returns the value UNKNOWN, if the encoding type cannot be determined.

Pragmas

none

Exception

If the audio plug-in raises an exception when calling this method, then it raises an AUDIO_PLUGIN_EXCEPTION exception.

Example

Call the format plug-in to read the actual encoding embedded in the stored audio data:

```
DECLARE
```

```
  obj ORDSYS.ORDAudio;
  ctx RAW(4000) :=NULL;
BEGIN
  SELECT aud INTO obj FROM TAUD  WHERE N=1;
  DBMS_OUTPUT.PUT_LINE('getting audio encoding');
  DBMS_OUTPUT.PUT_LINE('----------------------');
  DBMS_OUTPUT.PUT_LINE(obj.getEncoding(ctx));
  EXCEPTION
  WHEN ORDSYS.ORDAudioExceptions.AUDIO_PLUGIN_EXCEPTION THEN
  DBMS_OUTPUT.PUT_LINE('AUDIO_PLUGIN_EXCEPTION caught');
END;
/
```

setNumberOfChannels( ) Method

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

Calling this method implicitly calls the setUpdateTime( ) method.

Pragmas

none

Exception

If the value for the knownNumberOfChannels parameter is NULL, then calling this method raises a NULL_INPUT_VALUE exception.

Example

See the example in the setFormat( ) Method on page 3-76.

getNumberOfChannels Method

Format

```
getNumberOfChannels RETURN INTEGER;
```

Description

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

Parameters

none

Usage

none.

Pragmas

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

Exception

none

Example

See the example in the setFormat( ) Method on page 3-76.

getNumberOfChannels( ) Method

Format

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

Description

Calls the format plug-in to read the number of audio channels embedded in the stored audio data.

Parameters

ctx
The format plug-in context information.

Usage

The number of audio channels information is available from the header of the formatted audio data.

If the format found in the object is NULL, then the getNumberOfChannels( ) method uses the default format plug-in to read the audio data to determine the number of channels; otherwise, it uses the plug-in specified by the format.

Audio number of channels information can be extracted from the audio data itself. You can extend support to a format that is not understood by the ORDAudio object by implementing an ORDPLUGINS.ORDX_<format>_AUDIO package that supports that format. See Section 2.3.13 for more information.

Pragmas

none

Exception

If the audio plug-in raises an exception when calling this method, then it raises an AUDIO_PLUGIN_EXCEPTION exception.

Example

Call the format plug-in to read the actual number of audio channels embedded in the stored audio data:

```
DECLARE
  obj ORDSYS.ORDAudio;
  ctx RAW(4000) :=NULL;
BEGIN
  SELECT aud INTO obj FROM TAUD  WHERE N=1;
  DBMS_OUTPUT.PUT_LINE('getting audio channels');
  DBMS_OUTPUT.PUT_LINE('-------------------------');
  DBMS_OUTPUT.PUT_LINE(TO_CHAR(obj.getNumberOfChannels(ctx)));
  EXCEPTION
  WHEN ORDSYS.ORDAudioExceptions.AUDIO_PLUGIN_EXCEPTION THEN
  DBMS_OUTPUT.PUT_LINE('AUDIO_PLUGIN_EXCEPTION caught');
END;
/
```

setSamplingRate( ) Method

Format

```
setSamplingRate(knownSamplingRate IN INTEGER);
```

Description

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

Parameters

knownSamplingRate
A known sampling rate.

Usage

Calling this method implicitly calls the setUpdateTime( ) method.

Pragmas

none

Exception

If the value for the knownSamplingRate parameter is NULL, then calling this method raises a NULL_INPUT_VALUE exception.

Example

See the example in the setFormat( ) Method on page 3-76.

getSamplingRate Method

Format

```
getSamplingRate IN INTEGER;
```

Description

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

Parameters

none

Usage

none

Pragmas

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

Exception

none

Example

See the example in the setFormat( ) Method on page 3-76.

getSamplingRate( ) Method

Format

```
getSamplingRate(ctx IN OUT INTEGER);
```

Description

Calls the format plug-in to read the sampling rate embedded in the stored audio data.

Parameters

ctx
The format plug-in context information.

Usage

The audio sampling rate information is available from the header of the formatted audio data.

If the format found in the object is NULL, then the getSamplingRate( ) method uses the default format plug-in to read the audio data to determine the sampling rate; otherwise, it uses the plug-in specified by the format.

Audio sampling rate information can be set to a known sampling rate for the audio data. 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 Section 2.3.13 for more information.

Pragmas

none

Exception

If the audio plug-in raises an exception when calling this method, then it raises an AUDIO_PLUGIN_EXCEPTION exception.

Example

Return the sampling rate for audio data stored in the database:

```
DECLARE
```

```
  obj ORDSYS.ORDAudio;
  ctx RAW(4000) :=NULL;
BEGIN
  SELECT aud INTO obj FROM TAUD  WHERE N=1;
  DBMS_OUTPUT.PUT_LINE('getting sampling rate');
  DBMS_OUTPUT.PUT_LINE('---------------------');
  DBMS_OUTPUT.PUT_LINE(TO_CHAR(obj.getSamplingRate(ctx))||' KHz');
  EXCEPTION
  WHEN ORDSYS.ORDAudioExceptions.AUDIO_PLUGIN_EXCEPTION THEN
  DBMS_OUTPUT.PUT_LINE('AUDIO_PLUGIN_EXCEPTION caught');
END;
/
```

setSampleSize( ) Method

Format

```
setSampleSize(knownSampleSize IN INTEGER);
```

Description

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

Parameters

knownSampleSize
A known sample size.

Usage

Calling this method implicitly calls the setUpdateTime( ) method.

Pragmas

none

Exception

If the value for the knownSampleSize parameter is NULL, then calling this method raises a NULL_INPUT_VALUE exception.

Example

See the example in the setFormat( ) Method on page 3-76.

getSampleSize Method

Format

```
getSampleSize RETURN INTEGER;
```

Description

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

Parameters

none

Usage

none

Pragmas

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

Exception

none

Example

See the example in the setFormat( ) Method on page 3-76.

getSampleSize( ) Method

Format

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

Description

Calls the format plug-in to read the sample size embedded in the stored audio data.

Parameters

ctx
The format plug-in context information.

Usage

The audio sample size information is available from the header of the formatted audio data.

If the format found in the object is NULL, then the getSampleSize( ) method uses the default format plug-in to read the audio data to determine the sample size format; otherwise, it uses the plug-in specified by the format.

Audio sample size 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 Section 2.3.13 for more information.

Pragmas

none

Exception

If the audio plug-in raises an exception when calling this method, then it raises an AUDIO_PLUGIN_EXCEPTION exception.

Example

Return the sample size for audio data stored in the database:

```
DECLARE
  obj ORDSYS.ORDAudio;
```

```
  ctx RAW(4000) :=NULL;
BEGIN
  SELECT aud INTO obj FROM TAUD  WHERE N=1;
  DBMS_OUTPUT.PUT_LINE('getting sampling size');
  DBMS_OUTPUT.PUT_LINE('---------------------');
  DBMS_OUTPUT.PUT_LINE(TO_CHAR(obj.getSampleSize(ctx))||' bits');
  EXCEPTION
  WHEN ORDSYS.ORDAudioExceptions.AUDIO_PLUGIN_EXCEPTION THEN
  DBMS_OUTPUT.PUT_LINE('AUDIO_PLUGIN_EXCEPTION caught');
END;
/
```

setCompressionType( ) Method

Format

```
setCompressionType(knownCompressionType IN VARCHAR2);
```

Description

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

Parameters

knownCompressionType
A known compression type.

Usage

Calling this method implicitly calls the setUpdateTime( ) method.

Pragmas

none

Exception

If the value for the knownCompressionType parameter is NULL, then calling this method raises a NULL_INPUT_VALUE exception.

Example

See the example in the setFormat( ) Method on page 3-76.

getCompressionType Method

Format

```
getCompressionType RETURN VARCHAR2;
```

Description

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

Parameters

none

Usage

none

Pragmas

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

Exception

none

Example

See the example in the setFormat( ) Method on page 3-76.

getCompressionType( ) Method

Format

```
getCompressionType(ctx IN OUT RAW) RETURN VARCHAR2;
```

Description

Calls the format plug-in to read the compression type embedded in the stored audio data.

Parameters

ctx
The format plug-in context information.

Usage

The audio compression type information is available from the header of the formatted audio data.

If the format found in the object is NULL, then the getCompressionType( ) method uses the default format plug-in to read the audio data to determine the compression type; otherwise, it uses your user-defined format plug-in.

Audio compression type 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 Section 2.3.13 for more information.

Pragmas

none

Exception

If the audio plug-in raises an exception when calling this method, then it raises an AUDIO_PLUGIN_EXCEPTION exception.

Example

Return the type of compression used for audio data stored in the database:

```
DECLARE
```

```
  obj ORDSYS.ORDAudio;
  ctx RAW(4000) :=NULL;
BEGIN
  SELECT aud INTO obj FROM TAUD  WHERE N=1;
  DBMS_OUTPUT.PUT_LINE('getting compression type ');
  DBMS_OUTPUT.PUT_LINE('---------------------');
  DBMS_OUTPUT.PUT_LINE(obj.getCompressionType(ctx)|| ' ');
  EXCEPTION
        WHEN ORDSYS.ORDSourceExceptions.METHOD_NOT_SUPPORTED THEN
         DBMS_OUTPUT.put_line('Source METHOD_NOT_SUPPORTED caught');
        WHEN ORDSYS.ORDSourceExceptions.SOURCE_PLUGIN_EXCEPTION THEN
         DBMS_OUTPUT.put_line('SOURCE PLUGIN EXCEPTION caught');
        WHEN ORDSYS.ORDAudioExceptions.METHOD_NOT_SUPPORTED THEN
         DBMS_OUTPUT.put_line('AUDIO METHOD_NOT_SUPPORTED EXCEPTION caught');
        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;
/
```

setAudioDuration( ) Method

Format

```
setAudioDuration(knownAudioDuration IN INTEGER);
```

Description

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

Parameters

knownAudioDuration
A known audio duration.

Usage

Calling this method implicitly calls the setUpdateTime( ) method.

Pragmas

none

Exception

If the value for the knownAudioDuration parameter is NULL, then calling this method raises a NULL_INPUT_VALUE exception.

Example

See the example in the setFormat( ) Method on page 3-76.

getAudioDuration Method

Format

```
getAudioDuration RETURN INTEGER;
```

Description

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

Parameters

none

Usage

none

Pragmas

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

Exception

none

Example

See the example in the setFormat( ) Method on page 3-76.

getAudioDuration( ) Method

Format

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

Description

Calls the format plug-in to read the audio duration embedded in the stored audio data.

Parameters

ctx
The format plug-in context information.

Usage

The audio duration information is available from the header of the formatted audio data.

If the format found in the object is NULL, then the getAudioDuration( ) method uses the default format plug-in to read the audio data to determine the audio duration; otherwise, it uses your user-defined format plug-in.

Audio duration 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 Section 2.3.13 for more information.

Pragmas

none

Exception

If the audio plug-in raises an exception when calling this method, then it raises an AUDIO_PLUGIN_EXCEPTION exception.

Example

Return the duration or time to play the audio data stored in the database:

```
DECLARE
```

```
  obj ORDSYS.ORDAudio;
  ctx RAW(4000) :=NULL;
BEGIN
  SELECT aud INTO obj FROM TAUD  WHERE N=1 for update;
  DBMS_OUTPUT.PUT_LINE('getting audio duration');
  DBMS_OUTPUT.PUT_LINE('----------------------');
  obj.setFormat('WAVE');
  DBMS_OUTPUT.PUT_LINE(obj.getAudioDuration(ctx));
  update taud set aud = obj where n =1;
  EXCEPTION
        WHEN ORDSYS.ORDSourceExceptions.METHOD_NOT_SUPPORTED THEN
         DBMS_OUTPUT.put_line('Source METHOD_NOT_SUPPORTED caught');
        WHEN ORDSYS.ORDSourceExceptions.SOURCE_PLUGIN_EXCEPTION THEN
         DBMS_OUTPUT.put_line('SOURCE PLUGIN EXCEPTION caught');
        WHEN ORDSYS.ORDAudioExceptions.METHOD_NOT_SUPPORTED THEN
         DBMS_OUTPUT.put_line('AUDIO METHOD_NOT_SUPPORTED EXCEPTION caught');
        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;
/
```

setKnownAttributes( ) Method

Format

```
setKnownAttributes(
                    knownFormat           IN VARCHAR2,
                    knownEncoding         IN VARCHAR2,
                    knownNumberOfChannels IN INTEGER,
                    knownSamplingRate     IN INTEGER,
                    knownSamleSize        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

Calling this method implicitly calls the setUpdateTime( ) method.

Pragmas

none

Exception

none

Example

Set the known attributes for the audio data.

```
DECLARE
  obj ORDSYS.ORDAudio;
BEGIN
  select aud into obj from TAUD where N =1 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));
  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;
/
```

setProperties( ) Method

Format

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

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 the following attributes of the audio data: format, encoding type, number of channels, sampling rate, and sample size.

Parameters

ctx
The format plug-in context information.

Usage

If the format 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

Exception

If the audio plug-in raises an exception when calling this method, then it raises an AUDIO_PLUGIN_EXCEPTION exception.

Example

Set the property information for known audio attributes:

```
DECLARE
  obj ORDSYS.ORDAudio;
  ctx RAW(4000) :=NULL;
BEGIN
  select aud into obj from TAUD where N =1 for update;
  obj.setProperties(ctx);
  --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  TAUD set aud = obj where  N =1 ;
  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;
/
```

checkProperties( ) Method

Format

```
checkProperties(ctx IN OUT RAW) RETURN BOOLEAN;
```

Description

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

Parameters

ctx
The format plug-in context information.

Usage

If 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.

Pragmas

none

Exception

If the audio plug-in raises an exception when calling this method, then it raises an AUDIO_PLUGIN_EXCEPTION exception.

Example

Check property information for known audio attributes:

```
DECLARE
  obj ORDSYS.ORDAudio;
  ctx RAW(4000) :=NULL;
BEGIN
  select aud into obj from TAUD where N =1 for update;
  if( obj.checkProperties(ctx) =  TRUE ) then
  DBMS_OUTPUT.put_line('true');
  else
  DBMS_OUTPUT.put_line('false');
```

```
  end if;
  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;
/
```

getAttribute( ) Method

Format

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

Description

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

Parameters

ctx
The format plug-in context information.

name
The name of the attribute.

Usage

The audio data attributes are available from the header of the formatted audio data.

If the format is set to NULL, then the getAttribute( ) method uses the default format plug-in; otherwise, it uses your user-defined format plug-in.

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 Section 2.3.13 for more information.

Pragmas

none

Exception

If the audio plug-in raises an exception when calling this method, then it raises an AUDIO_PLUGIN_EXCEPTION exception.

Example

Return information for the specified audio attribute for audio data stored in the database:

```
DECLARE
  obj ORDSYS.ORDAudio;
  res VARCHAR2(4000);
  ctx RAW(4000) :=NULL;
BEGIN
  SELECT aud INTO obj FROM TAUD  WHERE N=1;
  DBMS_OUTPUT.PUT_LINE('getting audio sample size');
  DBMS_OUTPUT.PUT_LINE('---------------------');
  res := obj.getAttribute(ctx,'sample_size');
  EXCEPTION
        WHEN ORDSYS.ORDSourceExceptions.METHOD_NOT_SUPPORTED THEN
         DBMS_OUTPUT.put_line('Source METHOD_NOT_SUPPORTED caught');
        WHEN ORDSYS.ORDSourceExceptions.SOURCE_PLUGIN_EXCEPTION THEN
         DBMS_OUTPUT.put_line('SOURCE PLUGIN EXCEPTION caught');
        WHEN ORDSYS.ORDAudioExceptions.METHOD_NOT_SUPPORTED THEN
         DBMS_OUTPUT.put_line('AUDIO METHOD_NOT_SUPPORTED EXCEPTION caught');
        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;
/
```

getAllAttributes( ) Method

Format

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

Description

Returns a formatted string for convenient client access. For natively supported formats, the string includes the following list of audio data attributes separated by a comma (','): FileFormat, Encoding, NumberOfChannels, SamplingRate, and SampleSize. For user-defined formats, the string is defined by the format plug-in.

Parameters

ctx
The format plug-in context information.

attributes
The attributes.

Usage

These audio data attributes are available from the header of the formatted audio data.

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

Audio data attribute information can be extracted from the audio data itself. You can extend support to a format that is not understood by the ORDAudio object by implementing an ORDPLUGINS.ORDX_<format>_AUDIO package that supports this format. See Section 2.3.13 for more information.

Pragmas

none

Exception

If the audio plug-in raises an exception when calling this method, then it raises an AUDIO_PLUGIN_EXCEPTION exception.

Example

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

```
DECLARE
  obj ORDSYS.ORDAudio;
  tempLob   CLOB;
  ctx RAW(4000) :=NULL;
BEGIN
  SELECT aud INTO obj FROM TAUD  WHERE N=1;
  DBMS_OUTPUT.PUT_LINE('getting comma separated list of all attribs');
  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));
  EXCEPTION
  WHEN ORDSYS.ORDAudioExceptions.AUDIO_PLUGIN_EXCEPTION THEN
  DBMS_OUTPUT.PUT_LINE('ORDAudioExceptions.AUDIO_PLUGIN_EXCEPTION caught');
END;
/
```

3.2.9 ORDAudio Methods Associated with Processing Audio Data

This section presents reference information on the ORDAudio methods associated with processing audio data.

processAudioCommand( ) Method

Format

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

Description

Allows you to send a command and related arguments to the format plug-in for processing. This method is only supported for user-defined format plug-ins.

Parameters

ctx
The format plug-in context information.

cmd
Any command recognized by the format plug-in.

arguments
The arguments of the command.

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

Usage

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.

If the format is set to NULL, then the processAudioCommand( ) 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 ORDAudio object by preparing an ORDPLUGINS.ORDX_<format>_AUDIO package that supports that format. See Section 2.3.13 for more information.

Pragmas

none

Exception

If the audio plug-in raises an exception when calling this method, then it raises an AUDIO_PLUGIN_EXCEPTION exception.

Example

Process a set of commands:

```
DECLARE
  obj ORDSYS.ORDAudio;
  res RAW(4000);
  result RAW(4000);
  command VARCHAR(4000);
  argList VARCHAR(4000);
  ctx RAW(4000) :=NULL;
BEGIN
  select aud into obj from TAUD where N =1 for UPDATE;
  -- assign command
  -- assign argList
  res := obj.processAudioCommand (ctx, command, argList, result);
  UPDATE TAUD SET aud=obj WHERE N=1 ;
  COMMIT;
  EXCEPTION
        WHEN ORDSYS.ORDSourceExceptions.METHOD_NOT_SUPPORTED THEN
         DBMS_OUTPUT.put_line('Source METHOD_NOT_SUPPORTED caught');
        WHEN ORDSYS.ORDSourceExceptions.SOURCE_PLUGIN_EXCEPTION THEN
         DBMS_OUTPUT.put_line('SOURCE PLUGIN EXCEPTION caught');
        WHEN ORDSYS.ORDAudioExceptions.METHOD_NOT_SUPPORTED THEN
         DBMS_OUTPUT.put_line('AUDIO METHOD_NOT_SUPPORTED EXCEPTION caught');
        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;
/
```

3.3 Packages or PL/SQL Plug-ins

This section presents reference information on the packages or PL/SQL plug-ins provided.

3.3.1 ORDPLUGINS.ORDX_DEFAULT_AUDIO Package

Use the following provided ORDPLUGINS.ORDX_DEFAULT_AUDIO package as a guide in developing your own ORDPLUGINS.ORDX_<format>_AUDIO audio format package.

```
CREATE OR REPLACE PACKAGE ORDX_DEFAULT_AUDIO
authid current_user
AS
--AUDIO ATTRIBUTES ACCESSORS
FUNCTION getFormat(ctx IN OUT RAW, obj IN ORDSYS.ORDAudio)
                RETURN VARCHAR2;
FUNCTION getEncoding(ctx IN OUT RAW, obj IN ORDSYS.ORDAudio)
                RETURN VARCHAR2;
FUNCTION getNumberOfChannels(ctx IN OUT RAW, obj IN ORDSYS.ORDAudio)
                RETURN INTEGER;
FUNCTION getSamplingRate(ctx IN OUT RAW, obj IN ORDSYS.ORDAudio)
                RETURN INTEGER;
FUNCTION getSampleSize(ctx IN OUT RAW, obj IN ORDSYS.ORDAudio)
                RETURN INTEGER;
FUNCTION getCompressionType(ctx IN OUT RAW, obj IN ORDSYS.ORDAudio)
                RETURN VARCHAR2;
FUNCTION getAudioDuration(ctx IN OUT RAW, obj IN ORDSYS.ORDAudio)
                RETURN INTEGER;
PROCEDURE setProperties(ctx IN OUT RAW, obj IN OUT NOCOPY ORDSYS.ORDAudio);
FUNCTION checkProperties(ctx IN OUT RAW, obj IN OUT ORDSYS.ORDAudio) RETURN
NUMBER;
FUNCTION getAttribute(ctx IN OUT RAW,
                      obj IN ORDSYS.ORDAudio,
                      name IN VARCHAR2) RETURN VARCHAR2;
PROCEDURE getAllAttributes(ctx IN OUT RAW,
                           obj IN ORDSYS.ORDAudio,
                           attributes IN OUT NOCOPY CLOB);
--AUDIO PROCESSING METHODS
FUNCTION processCommand(ctx       IN OUT RAW,
                        obj       IN OUT NOCOPY ORDSYS.ORDAudio,
                        cmd       IN VARCHAR2,
                        arguments IN VARHAR2,
                        result    OUT RAW)
        RETURN RAW;
```

```
PRAGMA RESTRICT_REFERENCES(getFormat, WNDS, WNPS, RNDS, RNPS);
PRAGMA RESTRICT_REFERENCES(getEncoding, WNDS, WNPS, RNDS, RNPS);
PRAGMA RESTRICT_REFERENCES(getNumberOfChennels, WNDS, WNPS, RNDS, RNPS);
PRAGMA RESTRICT_REFERENCES(getSamplingRate, WNDS, WNPS, RNDS, RNPS);
PRAGMA RESTRICT_REFERENCES(getSampleSize, WNDS, WNPS, RNDS, RNPS);
PRAGMA RESTRICT_REFERENCES(getCompressionType, WNDS, WNPS, RNDS, RNPS);
PRAGMA RESTRICT_REFERENCES(getAttribute, WNDS, WNPS, RNDS, RNPS);
PRAGMA RESTRICT_REFERENCES(getAudioDuration, WNDS, WNPS, RNDS, RNPS);

END;
/
```

Table 3–1 shows the methods supported in the ORDPLUGINS.ORDX_DEFAULT_AUDIO package and the exceptions raised if you call a method that is not supported.

Table 3–1 Methods Supported in the ORDPLUGINS.ORDX_DEFAULT_AUDIO Package

Name of Method	Level of Support
getFormat	Supported; if the source is local, get the attribute and return the file format, but if the source is NULL, raise an ORDSYS.ORDSourceExceptions.EMPTY_SOURCE exception; otherwise, if the source is external, raise an ORDSYS.ORDAudioExceptions.UNSUPPORTED_DATA_SOURCE exception.
getEncoding	Supported; if the source is local, get the attribute and return the encoding, but if the source is NULL, raise an ORDSYS.ORDSourceExceptions.EMPTY_SOURCE exception; otherwise, if the source is external, raise an ORDSYS.ORDAudioExceptions.UNSUPPORTED_DATA_SOURCE exception.
getNumberOfChannels	Supported; if the source is local, get the attribute and return the number of channels, but if the source is NULL, raise an ORDSYS.ORDSourceExceptions.EMPTY_SOURCE exception; otherwise, if the source is external, raise an ORDSYS.ORDAudioExceptions.UNSUPPORTED_DATA_SOURCE exception.
getSamplingRate	Supported; if the source is local, get the attribute and return the sampling rate, but if the source is NULL, raise an ORDSYS.ORDSourceExceptions.EMPTY_SOURCE exception; otherwise, if the source is external, raise an ORDSYS.ORDAudioExceptions.UNSUPPORTED_DATA_SOURCE exception.
getSampleSize	Supported; if the source is local, get the attribute and return the sample size, but if the source is NULL, raise an ORDSYS.ORDSourceExceptions.EMPTY_SOURCE exception; otherwise, if the source is external, raise an ORDSYS.ORDAudioExceptions.UNSUPPORTED_DATA_SOURCE exception.

Table 3–1 Methods Supported in the ORDPLUGINS.ORDX_DEFAULT_AUDIO Package(Cont.)

Name of Method	Level of Support
getCompressionType	Not supported - raises exceptions: METHOD_NOT_SUPPORTED and AUDIO_PLUGIN_EXCEPTION.
getAudioDuration	Not supported - raises exceptions: METHOD_NOT_SUPPORTED and AUDIO_PLUGIN_EXCEPTION.
setProperties	Supported; if the source is local, process the local data and set the properties, but if the source is NULL, raise an ORDSYS.ORDSourceExceptions.EMPTY_SOURCE exception; if the source is a BFILE, then process the BFILE and set the properties; if the source is neither local nor a BFILE, get the media content into a temporary LOB, process the data, and set the properties.
checkProperties	Supported; if the source is local, process the local data and check the properties, but if the source is NULL, raise an ORDSYS.ORDSourceExceptions.EMPTY_SOURCE exception; if the source is a BFILE, then process the BFILE and check the properties; if the source is neither local nor a BFILE, get the media content into a temporary LOB, process the data, and check the properties.
getAttribute	Not supported - raises exceptions: METHOD_NOT_SUPPORTED and AUDIO_PLUGIN_EXCEPTION.
getAllAttributes	Supported; if the source is local, get the attributes and return them, but if the source is NULL, raise an ORDSYS.ORDSourceExceptions.EMPTY_SOURCE exception; otherwise, if the source is external, raise an ORDSYS.ORDAudioExceptions.UNSUPPORTED_DATA_SOURCE exception.
processCommand	Not supported - raises exceptions: METHOD_NOT_SUPPORTED and AUDIO_PLUGIN_EXCEPTION.

3.3.2 ORDPLUGINS.ORDX_AUFF_AUDIO Package

The ORDPLUGINS.ORDX_AUFF_AUDIO package is provided and is used for the AUFF audio format specification.

```
CREATE OR REPLACE PACKAGE ORDX_AUFF_AUDIO
authid current_user
AS
--AUDIO ATTRIBUTES ACCESSORS
FUNCTION getFormat(ctx IN OUT RAW, obj IN ORDSYS.ORDAudio)
                RETURN VARCHAR2;
FUNCTION getEncoding(ctx IN OUT RAW, obj IN ORDSYS.ORDAudio)
                RETURN VARCHAR2;
FUNCTION getNumberOfChannels(ctx IN OUT RAW, obj IN ORDSYS.ORDAudio)
                RETURN INTEGER;
FUNCTION getSamplingRate(ctx IN OUT RAW, obj IN ORDSYS.ORDAudio)
                RETURN INTEGER;
FUNCTION getSampleSize(ctx IN OUT RAW, obj IN ORDSYS.ORDAudio)
```

```
                RETURN INTEGER;
FUNCTION getCompressionType(ctx IN OUT RAW, obj IN ORDSYS.ORDAudio)
                RETURN VARCHAR2;
FUNCTION getAudioDuration(ctx IN OUT RAW, obj IN ORDSYS.ORDAudio)
                RETURN INTEGER;
PROCEDURE setProperties(ctx IN OUT RAW, obj IN OUT NOCOPY ORDSYS.ORDAudio);
FUNCTION checkProperties(ctx IN OUT RAW, obj IN OUT ORDSYS.ORDAudio)
         RETURN NUMBER;
FUNCTION getAttribute(ctx IN OUT RAW,
                      obj IN ORDSYS.ORDAudio,
                      name IN VARCHAR2) RETURN VARCHAR2;
PROCEDURE getAllAttributes(ctx IN OUT RAW,
                           obj IN ORDSYS.ORDAudio,
                           attributes IN OUT NOCOPY CLOB);
--AUDIO PROCESSING METHODS
FUNCTION processCommand(ctx       IN OUT RAW,
                        obj       IN OUT NOCOPY ORDSYS.ORDAudio,
                        cmd       IN VARCHAR2,
                        arguments IN VARHAR2,
                        result    OUT RAW)
         RETURN RAW;

PRAGMA RESTRICT_REFERENCES(getFormat, WNDS, WNPS, RNDS, RNPS);
PRAGMA RESTRICT_REFERENCES(getEncoding, WNDS, WNPS, RNDS, RNPS);
PRAGMA RESTRICT_REFERENCES(getNumberOfChennels, WNDS, WNPS, RNDS, RNPS);
PRAGMA RESTRICT_REFERENCES(getSamplingRate, WNDS, WNPS, RNDS, RNPS);
PRAGMA RESTRICT_REFERENCES(getSampleSize, WNDS, WNPS, RNDS, RNPS);
PRAGMA RESTRICT_REFERENCES(getCompressionType, WNDS, WNPS, RNDS, RNPS);
PRAGMA RESTRICT_REFERENCES(getAttribute, WNDS, WNPS, RNDS, RNPS);
PRAGMA RESTRICT_REFERENCES(getAudioDuration, WNDS, WNPS, RNDS, RNPS);

END;
/
```

Table 3–2 shows the methods supported in the ORDPLUGINS.ORDX_AUFF_AUDIO package and the exceptions raised if you call a method that is not supported.

Table 3–2 Methods Supported in the ORDPLUGINS.ORDX_AUFF_AUDIO Package

Name of Method	Level of Support
getFormat	Supported; if the source is local, get the attribute and return the file format, but if the source is NULL, raise an ORDSYS.ORDSourceExceptions.EMPTY_SOURCE exception; otherwise, if the source is external, raise an ORDSYS.ORDAudioExceptions.UNSUPPORTED_DATA_SOURCE exception.
getEncoding	Supported; if the source is local, get the attribute and return the encoding, but if the source is NULL, raise an ORDSYS.ORDSourceExceptions.EMPTY_SOURCE exception; otherwise, if the source is external, raise an ORDSYS.ORDAudioExceptions.UNSUPPORTED_DATA_SOURCE exception.
getNumberOfChannels	Supported; if the source is local, get the attribute and return the number of channels, but if the source is NULL, raise an ORDSYS.ORDSourceExceptions.EMPTY_SOURCE exception; otherwise, if the source is external, raise an ORDSYS.ORDAudioExceptions.UNSUPPORTED_DATA_SOURCE exception.
getSamplingRate	Supported; if the source is local, get the attribute and return the sampling rate, but if the source is NULL, raise an ORDSYS.ORDSourceExceptions.EMPTY_SOURCE exception; otherwise, if the source is external, raise an ORDSYS.ORDAudioExceptions.UNSUPPORTED_DATA_SOURCE exception.
getSampleSize	Supported; if the source is local, get the attribute and return the sample size, but if the source is NULL, raise an ORDSYS.ORDSourceExceptions.EMPTY_SOURCE exception; otherwise, if the source is external, raise an ORDSYS.ORDAudioExceptions.UNSUPPORTED_DATA_SOURCE exception.
getCompressionType	Not supported - raises exceptions: METHOD_NOT_SUPPORTED and AUDIO_PLUGIN_EXCEPTION.
getAudioDuration	Not supported - raises exceptions: METHOD_NOT_SUPPORTED and AUDIO_PLUGIN_EXCEPTION.
setProperties	Supported; if the source is local, process the local data and set the properties, but if the source is NULL, raise an ORDSYS.ORDSourceExceptions.EMPTY_SOURCE exception; if the source is a BFILE, then process the BFILE and set the properties; if the source is neither local nor a BFILE, get the media content into a temporary LOB, process the data, and set the properties.

Table 3–2 Methods Supported in the ORDPLUGINS.ORDX_AUFF_AUDIO Package (Cont.)

Name of Method	Level of Support
checkProperties	Supported; if the source is local, process the local data and check the properties, but if the source is NULL, raise an ORDSYS.ORDSourceExceptions.EMPTY_SOURCE exception; if the source is a BFILE, then process the BFILE and check the properties; if the source is neither local nor a BFILE, get the media content into a temporary LOB, process the data, and check the properties.
getAttribute	Not supported - raises exceptions: METHOD_NOT_SUPPORTED and AUDIO_PLUGIN_EXCEPTION.
getAllAttributes	Supported; if the source is local, get the attributes and return them, but if the source is NULL, raise an ORDSYS.ORDSourceExceptions.EMPTY_SOURCE exception; otherwise, if the source is external, raise an ORDSYS.ORDAudioExceptions.UNSUPPORTED_DATA_SOURCE exception.
processCommand	Not supported - raises exceptions: METHOD_NOT_SUPPORTED and AUDIO_PLUGIN_EXCEPTION.

3.3.3 ORDPLUGINS.ORDX_AIFF_AUDIO Package

The ORDPLUGINS.ORDX_AIFF_AUDIO package is provided and is used for the AIFF audio format specification.

```
CREATE OR REPLACE PACKAGE ORDX_AIFF_AUDIO
authid current_user
AS
--AUDIO ATTRIBUTES ACCESSORS
FUNCTION getFormat(ctx IN OUT RAW, obj IN ORDSYS.ORDAudio)
                RETURN VARCHAR2;
FUNCTION getEncoding(ctx IN OUT RAW, obj IN ORDSYS.ORDAudio)
                RETURN VARCHAR2;
FUNCTION getNumberOfChannels(ctx IN OUT RAW, obj IN ORDSYS.ORDAudio)
                RETURN INTEGER;
FUNCTION getSamplingRate(ctx IN OUT RAW, obj IN ORDSYS.ORDAudio)
                RETURN INTEGER;
FUNCTION getSampleSize(ctx IN OUT RAW, obj IN ORDSYS.ORDAudio)
                RETURN INTEGER;
FUNCTION getCompressionType(ctx IN OUT RAW, obj IN ORDSYS.ORDAudio)
                RETURN VARCHAR2;
FUNCTION getAudioDuration(ctx IN OUT RAW, obj IN ORDSYS.ORDAudio)
                RETURN INTEGER;
PROCEDURE setProperties(ctx IN OUT RAW, obj IN OUT NOCOPY ORDSYS.ORDAudio);
FUNCTION checkProperties(ctx IN OUT RAW, obj IN OUT ORDSYS.ORDAudio) RETURN
NUMBER;
FUNCTION getAttribute(ctx IN OUT RAW,
                      obj IN ORDSYS.ORDAudio,
```

```
                      name IN VARCHAR2) RETURN VARCHAR2;
PROCEDURE getAllAttributes(ctx IN OUT RAW,
                           obj IN ORDSYS.ORDAudio,
                           attributes IN OUT NOCOPY CLOB);
--AUDIO PROCESSING METHODS
FUNCTION processCommand(ctx       IN OUT RAW,
                        obj       IN OUT NOCOPY ORDSYS.ORDAudio,
                        cmd       IN VARCHAR2,
                        arguments IN VARHAR2,
                        result    OUT RAW)
         RETURN RAW;

PRAGMA RESTRICT_REFERENCES(getFormat, WNDS, WNPS, RNDS, RNPS);
PRAGMA RESTRICT_REFERENCES(getEncoding, WNDS, WNPS, RNDS, RNPS);
PRAGMA RESTRICT_REFERENCES(getNumberOfChennels, WNDS, WNPS, RNDS, RNPS);
PRAGMA RESTRICT_REFERENCES(getSamplingRate, WNDS, WNPS, RNDS, RNPS);
PRAGMA RESTRICT_REFERENCES(getSampleSize, WNDS, WNPS, RNDS, RNPS);
PRAGMA RESTRICT_REFERENCES(getCompressionType, WNDS, WNPS, RNDS, RNPS);
PRAGMA RESTRICT_REFERENCES(getAttribute, WNDS, WNPS, RNDS, RNPS);
PRAGMA RESTRICT_REFERENCES(getAudioDuration, WNDS, WNPS, RNDS, RNPS);

END;
/
```

Table 3–3 shows the methods supported in the ORDPLUGINS.ORDX_AIFF_AUDIO package and the exceptions raised if you call a method that is not supported.

Table 3–3 Methods Supported in the ORDPLUGINS.ORDX_AIFF_AUDIO Package

Name of Method	Level of Support
getFormat	Supported; if the source is local, get the attribute and return the file format, but if the source is NULL, raise an ORDSYS.ORDSourceExceptions.EMPTY_SOURCE exception; otherwise, if the source is external, raise an ORDSYS.ORDAudioExceptions.UNSUPPORTED_DATA_SOURCE exception.
getEncoding	Supported; if the source is local, get the attribute and return the encoding, but if the source is NULL, raise an ORDSYS.ORDSourceExceptions.EMPTY_SOURCE exception; otherwise, if the source is external, raise an ORDSYS.ORDAudioExceptions.UNSUPPORTED_DATA_SOURCE exception.

Table 3–3 Methods Supported in the ORDPLUGINS.ORDX_AIFF_AUDIO Package (Cont.)

Name of Method	Level of Support
getNumberOfChannels	Supported; if the source is local, get the attribute and return the number of channels, but if the source is NULL, raise an ORDSYS.ORDSourceExceptions.EMPTY_SOURCE exception; otherwise, if the source is external, raise an ORDSYS.ORDAudioExceptions.UNSUPPORTED_DATA_SOURCE exception.
getSamplingRate	Supported; if the source is local, get the attribute and return the sampling rate, but if the source is NULL, raise an ORDSYS.ORDSourceExceptions.EMPTY_SOURCE exception; otherwise, if the source is external, raise an ORDSYS.ORDAudioExceptions.UNSUPPORTED_DATA_SOURCE exception.
getSampleSize	Supported; if the source is local, get the attribute and return the sample size, but if the source is NULL, raise an ORDSYS.ORDSourceExceptions.EMPTY_SOURCE exception; otherwise, if the source is external, raise an ORDSYS.ORDAudioExceptions.UNSUPPORTED_DATA_SOURCE exception.
getCompressionType	Not supported - raises exceptions: METHOD_NOT_SUPPORTED and AUDIO_PLUGIN_EXCEPTION.
getAudioDuration	Not supported - raises exceptions: METHOD_NOT_SUPPORTED and AUDIO_PLUGIN_EXCEPTION.
setProperties	Supported; if the source is local, process the local data and set the properties, but if the source is NULL, raise an ORDSYS.ORDSourceExceptions.EMPTY_SOURCE exception; if the source is a BFILE, then process the BFILE and set the properties; if the source is neither local nor a BFILE, get the media content into a temporary LOB, process the data, and set the properties.
checkProperties	Supported; if the source is local, process the local data and check the properties, but if the source is NULL, raise an ORDSYS.ORDSourceExceptions.EMPTY_SOURCE exception; if the source is a BFILE, then process the BFILE and check the properties; if the source is neither local nor a BFILE, get the media content into a temporary LOB, process the data, and check the properties.
getAttribute	Not supported - raises exceptions: METHOD_NOT_SUPPORTED and AUDIO_PLUGIN_EXCEPTION.
getAllAttributes	Supported; if the source is local, get the attributes and return them, but if the source is NULL, raise an ORDSYS.ORDSourceExceptions.EMPTY_SOURCE exception; otherwise, if the source is external, raise an ORDSYS.ORDAudioExceptions.UNSUPPORTED_DATA_SOURCE exception.
processCommand	Not supported - raises exceptions: METHOD_NOT_SUPPORTED and AUDIO_PLUGIN_EXCEPTION.

3.3.4 ORDPLUGINS.ORDX_AIFC_AUDIO Package

The ORDPLUGINS.ORDX_AIFC_AUDIO package is provided and is used for the AIFC audio format specification.

```
CREATE OR REPLACE PACKAGE ORDX_AIFC_AUDIO
authid current_user
AS
--AUDIO ATTRIBUTES ACCESSORS
FUNCTION getFormat(ctx IN OUT RAW, obj IN ORDSYS.ORDAudio)
                RETURN VARCHAR2;
FUNCTION getEncoding(ctx IN OUT RAW, obj IN ORDSYS.ORDAudio)
                RETURN VARCHAR2;
FUNCTION getNumberOfChannels(ctx IN OUT RAW, obj IN ORDSYS.ORDAudio)
                RETURN INTEGER;
FUNCTION getSamplingRate(ctx IN OUT RAW, obj IN ORDSYS.ORDAudio)
                RETURN INTEGER;
FUNCTION getSampleSize(ctx IN OUT RAW, obj IN ORDSYS.ORDAudio)
                RETURN INTEGER;
FUNCTION getCompressionType(ctx IN OUT RAW, obj IN ORDSYS.ORDAudio)
                RETURN VARCHAR2;
FUNCTION getAudioDuration(ctx IN OUT RAW, obj IN ORDSYS.ORDAudio)
                RETURN INTEGER;
PROCEDURE setProperties(ctx IN OUT RAW, obj IN OUT NOCOPY ORDSYS.ORDAudio);
FUNCTION checkProperties(ctx IN OUT RAW, obj IN OUT ORDSYS.ORDAudio)
        RETURN NUMBER;
FUNCTION getAttribute(ctx IN OUT RAW,
                      obj IN ORDSYS.ORDAudio,
                      name IN VARCHAR2) RETURN VARCHAR2;
PROCEDURE getAllAttributes(ctx IN OUT RAW,
                           obj IN ORDSYS.ORDAudio,
                           attributes IN OUT NOCOPY CLOB);
--AUDIO PROCESSING METHODS
FUNCTION processCommand(ctx       IN OUT RAW,
                        obj       IN OUT NOCOPY ORDSYS.ORDAudio,
                        cmd       IN VARCHAR2,
                        arguments IN VARHAR2,
                        result    OUT RAW)
        RETURN RAW;

PRAGMA RESTRICT_REFERENCES(getFormat, WNDS, WNPS, RNDS, RNPS);
PRAGMA RESTRICT_REFERENCES(getEncoding, WNDS, WNPS, RNDS, RNPS);
PRAGMA RESTRICT_REFERENCES(getNumberOfChennels, WNDS, WNPS, RNDS, RNPS);
PRAGMA RESTRICT_REFERENCES(getSamplingRate, WNDS, WNPS, RNDS, RNPS);
PRAGMA RESTRICT_REFERENCES(getSampleSize, WNDS, WNPS, RNDS, RNPS);
PRAGMA RESTRICT_REFERENCES(getCompressionType, WNDS, WNPS, RNDS, RNPS);
```

```
PRAGMA RESTRICT_REFERENCES(getAttribute, WNDS, WNPS, RNDS, RNPS);
PRAGMA RESTRICT_REFERENCES(getAudioDuration, WNDS, WNPS, RNDS, RNPS);

END;
/
```

Table 3–4 shows the methods supported in the ORDPLUGINS.ORDX_AIFC_AUDIO package and the exceptions raised if you call a method that is not supported.

Table 3–4 Methods Supported in the ORDPLUGINS.ORDX_AIFC_AUDIO Package

Name of Method	Level of Support
getFormat	Supported; if the source is local, get the attribute and return the file format, but if the source is NULL, raise an ORDSYS.ORDSourceExceptions.EMPTY_SOURCE exception; otherwise, if the source is external, raise an ORDSYS.ORDAudioExceptions.UNSUPPORTED_DATA_SOURCE exception.
getEncoding	Supported; if the source is local, get the attribute and return the encoding, but if the source is NULL, raise an ORDSYS.ORDSourceExceptions.EMPTY_SOURCE exception; otherwise, if the source is external, raise an ORDSYS.ORDAudioExceptions.UNSUPPORTED_DATA_SOURCE exception.
getNumberOfChannels	Supported; if the source is local, get the attribute and return the number of channels, but if the source is NULL, raise an ORDSYS.ORDSourceExceptions.EMPTY_SOURCE exception; otherwise, if the source is external, raise an ORDSYS.ORDAudioExceptions.UNSUPPORTED_DATA_SOURCE exception.
getSamplingRate	Supported; if the source is local, get the attribute and return the sampling rate, but if the source is NULL, raise an ORDSYS.ORDSourceExceptions.EMPTY_SOURCE exception; otherwise, if the source is external, raise an ORDSYS.ORDAudioExceptions.UNSUPPORTED_DATA_SOURCE exception.
getSampleSize	Supported; if the source is local, get the attribute and return the sample size, but if the source is NULL, raise an ORDSYS.ORDSourceExceptions.EMPTY_SOURCE exception; otherwise, if the source is external, raise an ORDSYS.ORDAudioExceptions.UNSUPPORTED_DATA_SOURCE exception.
getCompressionType	Not supported - raises exceptions: METHOD_NOT_SUPPORTED and AUDIO_PLUGIN_EXCEPTION.
getAudioDuration	Not supported - raises exceptions: METHOD_NOT_SUPPORTED and AUDIO_PLUGIN_EXCEPTION.

Table 3–4 Methods Supported in the ORDPLUGINS.ORDX_AIFC_AUDIO Package (Cont.)

Name of Method	Level of Support
setProperties	Supported; if the source is local, process the local data and set the properties, but if the source is NULL, raise an ORDSYS.ORDSourceExceptions.EMPTY_SOURCE exception; if the source is a BFILE, then process the BFILE and set the properties; if the source is neither local nor a BFILE, get the media content into a temporary LOB, process the data, and set the properties.
checkProperties	Supported; if the source is local, process the local data and check the properties, but if the source is NULL, raise an ORDSYS.ORDSourceExceptions.EMPTY_SOURCE exception; if the source is a BFILE, then process the BFILE and check the properties; if the source is neither local nor a BFILE, get the media content into a temporary LOB, process the data, and check the properties.
getAttribute	Not supported - raises exceptions: METHOD_NOT_SUPPORTED and AUDIO_PLUGIN_EXCEPTION.
getAllAttributes	Supported; if the source is local, get the attributes and return them, but if the source is NULL, raise an ORDSYS.ORDSourceExceptions.EMPTY_SOURCE exception; otherwise, if the source is external, raise an ORDSYS.ORDAudioExceptions.UNSUPPORTED_DATA_SOURCE exception.
processCommand	Not supported - raises exceptions: METHOD_NOT_SUPPORTED and AUDIO_PLUGIN_EXCEPTION.

3.3.5 ORDPLUGINS.ORDX_WAVE_AUDIO Package

The ORDPLUGINS.ORDX_WAVE_AUDIO package is provided and is used for the WAVE audio format specification.

```
CREATE OR REPLACE PACKAGE ORDX_WAVE_AUDIO
authid current_user
AS
--AUDIO ATTRIBUTES ACCESSORS
FUNCTION getFormat(ctx IN OUT RAW, obj IN ORDSYS.ORDAudio)
                RETURN VARCHAR2;
FUNCTION getEncoding(ctx IN OUT RAW, obj IN ORDSYS.ORDAudio)
                RETURN VARCHAR2;
FUNCTION getNumberOfChannels(ctx IN OUT RAW, obj IN ORDSYS.ORDAudio)
                RETURN INTEGER;
FUNCTION getSamplingRate(ctx IN OUT RAW, obj IN ORDSYS.ORDAudio)
                RETURN INTEGER;
FUNCTION getSampleSize(ctx IN OUT RAW, obj IN ORDSYS.ORDAudio)
                RETURN INTEGER;
FUNCTION getCompressionType(ctx IN OUT RAW, obj IN ORDSYS.ORDAudio)
                RETURN VARCHAR2;
FUNCTION getAudioDuration(ctx IN OUT RAW, obj IN ORDSYS.ORDAudio)
```

```
                RETURN INTEGER;
PROCEDURE setProperties(ctx IN OUT RAW, obj IN OUT NOCOPY ORDSYS.ORDAudio);
FUNCTION checkProperties(ctx IN OUT RAW, obj IN OUT ORDSYS.ORDAudio)
         RETURN NUMBER;
FUNCTION getAttribute(ctx IN OUT RAW,
                      obj IN ORDSYS.ORDAudio,
                      name IN VARCHAR2) RETURN VARCHAR2;
PROCEDURE getAllAttributes(ctx IN OUT RAW,
                           obj IN ORDSYS.ORDAudio,
                           attributes IN OUT NOCOPY CLOB);
--AUDIO PROCESSING METHODS
FUNCTION processCommand(ctx       IN OUT RAW,
                        obj       IN OUT NOCOPY ORDSYS.ORDAudio,
                        cmd       IN VARCHAR2,
                        arguments IN VARHAR2,
                        result    OUT RAW)
         RETURN RAW;

PRAGMA RESTRICT_REFERENCES(getFormat, WNDS, WNPS, RNDS, RNPS);
PRAGMA RESTRICT_REFERENCES(getEncoding, WNDS, WNPS, RNDS, RNPS);
PRAGMA RESTRICT_REFERENCES(getNumberOfChennels, WNDS, WNPS, RNDS, RNPS);
PRAGMA RESTRICT_REFERENCES(getSamplingRate, WNDS, WNPS, RNDS, RNPS);
PRAGMA RESTRICT_REFERENCES(getSampleSize, WNDS, WNPS, RNDS, RNPS);
PRAGMA RESTRICT_REFERENCES(getCompressionType, WNDS, WNPS, RNDS, RNPS);
PRAGMA RESTRICT_REFERENCES(getAttribute, WNDS, WNPS, RNDS, RNPS);
PRAGMA RESTRICT_REFERENCES(getAudioDuration, WNDS, WNPS, RNDS, RNPS);

END;
/
```

Table 3–5 shows the methods supported in the ORDPLUGINS.ORDX_WAVE_AUDIO package and the exceptions raised if you call a method that is not supported.

Table 3–5 Methods Supported in the ORDPLUGINS.ORDX_WAVE_AUDIO Package

Name of Method	Level of Support
getFormat	Supported; if the source is local, get the attribute and return the file format, but if the source is NULL, raise an ORDSYS.ORDSourceExceptions.EMPTY_SOURCE exception; otherwise, if the source is external, raise an ORDSYS.ORDAudioExceptions.UNSUPPORTED_DATA_SOURCE exception.

Table 3–5 Methods Supported in the ORDPLUGINS.ORDX_WAVE_AUDIO Package (Cont.)

Name of Method	Level of Support
getEncoding	Supported; if the source is local, get the attribute and return the encoding, but if the source is NULL, raise an ORDSYS.ORDSourceExceptions.EMPTY_SOURCE exception; otherwise, if the source is external, raise an ORDSYS.ORDAudioExceptions.UNSUPPORTED_DATA_SOURCE exception.
getNumberOfChannels	Supported; if the source is local, get the attribute and return the number of channels, but if the source is NULL, raise an ORDSYS.ORDSourceExceptions.EMPTY_SOURCE exception; otherwise, if the source is external, raise an ORDSYS.ORDAudioExceptions.UNSUPPORTED_DATA_SOURCE exception.
getSamplingRate	Supported; if the source is local, get the attribute and return the sampling rate, but if the source is NULL, raise an ORDSYS.ORDSourceExceptions.EMPTY_SOURCE exception; otherwise, if the source is external, raise an ORDSYS.ORDAudioExceptions.UNSUPPORTED_DATA_SOURCE exception.
getSampleSize	Supported; if the source is local, get the attribute and return the sample size, but if the source is NULL, raise an ORDSYS.ORDSourceExceptions.EMPTY_SOURCE exception; otherwise, if the source is external, raise an ORDSYS.ORDAudioExceptions.UNSUPPORTED_DATA_SOURCE exception.
getCompressionType	Not supported - raises exceptions: METHOD_NOT_SUPPORTED and AUDIO_PLUGIN_EXCEPTION.
getAudioDuration	Not supported - raises exceptions: METHOD_NOT_SUPPORTED and AUDIO_PLUGIN_EXCEPTION.
setProperties	Supported; if the source is local, process the local data and set the properties, but if the source is NULL, raise an ORDSYS.ORDSourceExceptions.EMPTY_SOURCE exception; if the source is a BFILE, then process the BFILE and set the properties; if the source is neither local nor a BFILE, get the media content into a temporary LOB, process the data, and set the properties.
checkProperties	Supported; if the source is local, process the local data and check the properties, but if the source is NULL, raise an ORDSYS.ORDSourceExceptions.EMPTY_SOURCE exception; if the source is a BFILE, then process the BFILE and check the properties; if the source is neither local nor a BFILE, get the media content into a temporary LOB, process the data, and check the properties.
getAttribute	Not supported - raises exceptions: METHOD_NOT_SUPPORTED and AUDIO_PLUGIN_EXCEPTION.
getAllAttributes	Supported; if the source is local, get the attributes and return them, but if the source is NULL, raise an ORDSYS.ORDSourceExceptions.EMPTY_SOURCE exception; otherwise, if the source is external, raise an ORDSYS.ORDAudioExceptions.UNSUPPORTED_DATA_SOURCE exception.

Table 3–5 Methods Supported in the ORDPLUGINS.ORDX_WAVE_AUDIO Package (Cont.)

Name of Method	Level of Support
processCommand	Not supported - raises exceptions: METHOD_NOT_SUPPORTED and AUDIO_PLUGIN_EXCEPTION.

3.3.6 Extending *inter*Media to Support a New Audio Data Format

Extending *inter*Media to support a new audio data format consists of four steps:

1. Design your new audio data format.
2. Implement your new audio data format and name it, for example, ORDX_MY_AUDIO.SQL.
3. Install your new ORDX_MY_AUDIO.SQL plug-in in the ORDPLUGINS schema.
4. Grant EXECUTE privileges on your new plug-in, for example, ORDX_MY_AUDIO.SQL plug-in, to PUBLIC.

Section 2.1.14 briefly describes how to extend *inter*Media to support a new audio data format and describes the interface. A package body listing is provided in Example 3–1 to assist you in this operation. Add your variables to the places that say "--Your variables go here" and add your code to the places that say "--Your code goes here".

See Section E.1 for more information on installing your own audio format plug-in and running the sample scripts provided.

Example 3–1 Package Body Listing for Extending Support to a New Audio Data Format

```
CREATE OR REPLACE PACKAGE BODY ORDX_MY_AUDIO
AS
  --AUDIO ATTRIBUTES ACCESSORS
  FUNCTION  getFormat(ctx IN OUT RAW, obj IN ORDSYS.ORDAudio)
  RETURN VARCHAR2
  IS
--Your variables go here
  BEGIN
--Your code goes here
  END;
  FUNCTION  getEncoding(ctx IN OUT RAW, obj IN ORDSYS.ORDAudio)
  RETURN VARCHAR2
  IS
--Your variables go here
```

```
  BEGIN
--Your code goes here
  END;
  FUNCTION  getNumberOfChannels(ctx IN OUT RAW, obj IN ORDSYS.ORDAudio)
  RETURN INTEGER
  IS
--Your variables go here
  BEGIN
--Your code goes here
  END;
  FUNCTION  getSamplingRate(ctx IN OUT RAW, obj IN ORDSYS.ORDAudio)
  RETURN INTEGER
  IS
--Your variables go here
  BEGIN
--Your code goes here
  END;
  FUNCTION  getSampleSize(ctx IN OUT RAW, obj IN ORDSYS.ORDAudio)
  RETURN INTEGER
  IS
--Your variables go here
  BEGIN
--Your code goes here
  END;
  FUNCTION  getCompressionType(ctx IN OUT RAW, obj IN ORDSYS.ORDAudio)
  RETURN VARCHAR2
  IS
--Your variables go here
  BEGIN
--Your code goes here
  END;
  FUNCTION  getAudioDuration(ctx IN OUT RAW,
                             obj IN ORDSYS.ORDAudio)
  RETURN INTEGER
  IS
--Your variables go here
  BEGIN
--Your code goes here
  END;
  PROCEDURE setProperties(ctx IN OUT RAW, obj IN OUT NOCOPY ORDSYS.ORDAudio)
  IS
--Your variables go here
  BEGIN
--Your code goes here
  END;
```

```
  FUNCTION checkProperties(ctx IN OUT RAW, obj IN OUT ORDSYS.ORDAudio)
  RETURN NUMBER
  IS
--Your variables go here
  BEGIN
--Your code goes here
  END;
  FUNCTION  getAttribute(ctx IN OUT RAW,
                         obj IN ORDSYS.ORDAudio,
                         name IN VARCHAR2)
  RETURN VARCHAR2
  IS
--Your variables go here
  BEGIN
--Your code goes here
  END;
  PROCEDURE getAllAttributes(ctx IN OUT RAW,
                             obj IN ORDSYS.ORDAudio,
                             attributes IN OUT NOCOPY CLOB)
  IS
--Your variables go here
  BEGIN
--Your code goes here
  END;
  -- AUDIO PROCESSING METHODS
  FUNCTION  processCommand(
                                  ctx       IN OUT RAW,
                                  obj       IN OUT NOCOPY ORDSYS.ORDAudio,
                                  cmd       IN VARCHAR2,
                                  arguments IN VARCHAR2,
                                  result    OUT RAW)
  RETURN RAW
  IS
--Your variables go here
  BEGIN
--Your code goes here
  END;
END;
/
show errors;
```

4

ORDImage Reference Information

Oracle8*i* *inter*Media contains the following information about the ORDImage type:

- Object type -- see Section 4.1.
- Methods -- see Section 4.2.

The examples in this chapter assume that the test image tables EMP and OLD_IMAGE have been created and filled with data. These tables were created using the SQL statements described in Section 4.2.1.

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

Methods invoked at the ORDSource level that are handed off to the source plug-in for processing have ctx (RAW(4000)) 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 source.open( ) method. At this point, the source plug-in can initialize the context for this client. When processing is complete, the client should invoke the source.close( ) method.

Methods invoked from a source plug-in call have the first argument as ctx (RAW(4000)).

Note: In the current release, not all source plug-ins will use the ctx argument, but if you code as previously described, your application should work with any current or future source plug-in.

4.1 Object Types

Oracle8*i* *inter*Media describes the ORDImage object type, which supports the storage, management, and manipulation of image data.

ORDImage Object Type

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

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

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

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

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

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

  -- Methods associated with image property set and check operations
  MEMBER PROCEDURE setProperties,

  MEMBER PROCEDURE setProperties(description IN VARCHAR2),

  MEMBER FUNCTION checkProperties RETURN BOOLEAN,

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

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

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

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

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

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

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

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

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

-- Methods associated with the source attribute
MEMBER FUNCTION  getContent RETURN BLOB,
PRAGMA RESTRICT_REFERENCES(getContent, WNDS, WNPS, RNDS, RNPS),

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

MEMBER PROCEDURE deleteContent,

MEMBER PROCEDURE setSource(source_type     IN VARCHAR2,
                           source_location IN VARCHAR2,
                           source_name     IN VARCHAR2),
MEMBER FUNCTION  getSource RETURN VARCHAR2,
```

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

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

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

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

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

  -- Methods associated with image migration
  MEMBER PROCEDURE migrateFromORDImgB(old_object  ORDImgB),
  MEMBER PROCEDURE migrateFromORDImgF(old_object  ORDImgF)
);
```

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 *on-disk* image file in bytes.
- fileFormat: the file type or format in which the image data is stored (TIFF, JIFF, and so forth.).
- contentFormat: the type of image (monochrome, 8-bit grayscale, and so forth).
- compressionFormat: the compression algorithm used on the image data.
- mimeType: the MIME type information.

4.2 Methods

This section presents reference information on the Oracle8*i* *inter*Media methods used for image data manipulation. These methods are described in the following groupings:

ORDImage Methods Associated with copy Operations

- copy( ): creates a copy of an image in another ORDImage.

ORDImage Methods Associated with process Operations

- process( ): performs in-place image processing on an image stored in a BLOB.
- processCopy( ): performs image processing while copying an image to another ORDImage BLOB data type.

ORDImage Methods Associated with properties set and check Operations

- setProperties: fills in the attribute fields of an image for native image formats.
- setProperties( ): fills in the attribute fields of an image and includes a description parameter for foreign image formats. See the "setProperties( ) Method for Foreign Images" for a description of what a foreign image is.
- checkProperties: verifies the stored image attributes match the actual image.

ORDImage Methods Associated with image Attributes

- getHeight: returns the height of the image in pixels.
- getWidth: returns the width of the image in pixels.
- getContentLength: returns the size of the image in bytes.
- getFileFormat: returns the file type of an image.
- getContentFormat: returns the format of the image.
- getCompressionFormat: returns the type of compression used on the image.

ORDImage Methods Associated with the local Attribute

- setLocal: sets a flag to indicate that the data is stored locally in a BLOB.
- clearLocal: clears the flag to indicate that the data is stored externally.

- isLocal: returns TRUE if the data is stored locally in a BLOB or FALSE if the data is external.

ORDImage Methods Associated with the date Attribute

- getUpdateTime: returns the time when the image object was last updated.
- setUpdateTime( ): sets the update time for the image object.

ORDImage Methods Associated with the mimeType Attribute

- getMimeType: returns the MIME type of the stored image data.
- setMimeType( ): sets the MIME type of the stored image data.

ORDImage Methods Associated with the source Attribute

- getContent: returns the content of the local data.
- getBFILE: returns the external content as a BFILE.
- deleteContent: deletes the content of the local data.
- setSource( ): sets the source information to where external image data is to be found.
- getSource: returns a string containing complete information about the external data source formatted as a URL.
- getSourceType: returns the external source type of the image data.
- getSourceLocation: returns the external source location of the image data.
- getSourceName: returns the external source name of the image data.
- import( ): transfers data from an external data source (specified by calling setSourceInformation( )) to the local source (localData) within an Oracle database, setting the value of the local attribute to "1", meaning local, and updating the timestamp and image attributes.
- importFrom( ): transfers data from the specified external data source (source type, location, name) to the local source (localData) within an Oracle database, setting the value of the local attribute to "1", meaning local, and updating the timestamp and image attributes.
- export( ): transfers data from a local source (localData) within an Oracle database to the specified external data source, setting source information to parame-

ters supplied, setting the value of the local attribute to "0", meaning external, and leaving all attributes unchanged.

Note: Sources natively supported by *inter*Media are not writable and therefore do not support the export method. User-defined sources may support the export method.

ORDImage Methods Associated with Image Migration

- migrateFromORDImgB: copies old ORDImgB images to ORDImage objects.
- migrateFromORDImgF: copies old ORDImgF images to ORDImage objects.

4.2.1 Example Table Definitions

The methods described in this chapter show examples based on a test image table EMP. Refer to the EMP table definition that follows when reading through the examples in Section 4.2.2 through Section 4.2.9:

EMP Table Definition

```
CREATE TABLE emp (
  ename VARCHAR2(50),
  salary NUMBER,
  job VARCHAR2(50),
  department INTEGER,
  photo ORDSYS.ORDImage,
  large_photo ORDSYS.ORDImage);
DECLARE
  Image ORDSYS.ORDImage;
BEGIN
  INSERT INTO emp VALUES ('John Doe', 24000, 'Technical Writer', 123,
      ORDSYS.ORDImage(ORDSYS.ORDSource(empty_blob(),'file','ORDIMGDIR',
                                       'jdoe.gif',SYSDATE,0),
                      NULL,NULL,NULL,NULL,NULL,NULL,NULL),
      ORDSYS.ORDImage(ORDSYS.ORDSource(empty_blob(),'file','ORDIMGDIR',
                                       'jdoe.gif',SYSDATE,0),
                      NULL,NULL,NULL,NULL,NULL,NULL,NULL));
  SELECT large_photo INTO Image FROM emp
         WHERE ename = 'John Doe' FOR UPDATE;
  Image.setProperties;
  UPDATE emp SET large_photo = Image WHERE ename = 'John Doe';
  COMMIT;
END;
/
```

Refer to the EMP and OLD_IMAGES table definitions that follow when reading through the examples in Section 4.2.10.

EMP and OLD_IMAGES Table Definitions

```
CREATE TABLE emp (
  ename VARCHAR2(50),
  salary NUMBER,
  job VARCHAR2(50),
  department INTEGER,
  large_photo ORDSYS.ORDImage);
CREATE TABLE old_images (
  id NUMBER,
  imageb ORDSYS.ORDIMGB,
  imagef ORDSYS.ORDIMGF);
DECLARE
  blobimage ORDSYS.ORDIMGB;
  bfileimage ORDSYS.ORDIMGF;
BEGIN
  INSERT INTO old_images values
      (1, ORDSYS.ORDIMGB(empty_blob(),NULL,NULL,NULL,NULL,NULL,NULL),
       ORDSYS.ORDIMGF(bfilename('ORDIMGDIR','jdoe.gif'),
                      NULL,NULL,NULL,NULL,NULL,NULL));
  SELECT imageb, imagef INTO blobimage, bfileimage
     FROM old_images WHERE id = 1 FOR UPDATE;
  bfileimage.copyContent(blobimage.content);
  blobimage.setProperties;
  bfileimage.setProperties;
  UPDATE old_images SET imageb=blobimage, imagef=bfileimage WHERE id = 1;
  INSERT INTO emp values
    ( 'John Doe', 24000, 'Technical Writer', 123,
     ORDSYS.ORDImage(ORDSYS.ORDSource(empty_blob(),NULL,NULL,NULL,
                                      SYSDATE,1),
                  NULL,NULL,NULL,NULL,NULL,NULL,NULL));
  COMMIT;
end;
/
```

4.2.2 ORDImage Methods Associated with Copy Operations

This section presents reference information on the ORDImage method associated with the copy operation.

copy( ) Method

Format

```
copy(dest IN OUT ORDImage);
```

Description

Copies an image without changing it.

Parameters

dest
The destination of the new image.

Usage

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

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

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

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

Pragmas

none

Exception

If the destination source.localData attribute is not initialized, calling this method raises a NULL_LOCAL_DATA exception.

If the source.isLocal attribute value is 1 and the source.localData attribute value is NULL, calling this method raises a NULL_LOCAL_DATA exception.

Example

Create a copy of the image:

```
DECLARE
    Image_1 ORDSYS.ORDImage;
    Image_2 ORDSYS.ORDImage;
BEGIN
    SELECT photo, large_photo
       INTO Image_2, Image_1
       FROM emp
       WHERE ename = 'John Doe' FOR UPDATE;
    -- copy the data from Image_1 to Image_2
    Image_1.copy(Image_2);
    UPDATE emp SET photo = Image_2
        WHERE ename = 'John Doe';
END;
/
```

4.2.3 ORDImage Methods Associated with Process Operations

This section presents reference information on the ORDImage methods associated with the process operation.

process( ) Method

Format

```
process(command IN VARCHAR2);
```

Description

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

Parameters

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

Usage

You can change one or more of the image attributes shown in Table 4–1. Table 4–2 shows additional changes that can be made only to raw pixel and foreign images.

Table 4–1 Image Processing Operators

Operator Name	Usage	Values
compressionFormat	Compression type / format	JPEG, SUNRLE, BMPRLE, TARGARLE, LZW, LZWHDIFF, FAX3, FAX4, HUFFMAN3, Packbits, GIFLZW
compressionQuality	Compression quality	MAXCOMPRATIO, MAXINTEGRITY, LOWCOMP, MEDCOMP, HIGHCOMP
contentFormat	Image type / pixel / data format	MONOCHROME, 8BITGRAYSCALE, 8BITGREYSCALE, 8BITLUT, 24BITRGB
cut	Window to cut or crop (origin.x origin.y width height)	(Integer Integer Integer Integer) maximum value is 65535
fileFormat	File format of the image	BMPF, CALS, GIFF, JFIF, PICT, RASF, RPIX, TGAF, TIFF
fixedScale	Scale to a specific size in pixels (width, height)	(INTEGER INTEGER)

Table 4–1 Image Processing Operators(Cont.)

Operator Name	Usage	Values
maxScale	Scale to a specific size in pixels, while maintaining the aspect ratio (maxWidth, maxHeight)	(INTEGER INTEGER)
scale	Scale factor (for example, 0.5 or 2.0)	<FLOAT> positive
xScale	X-axis scale factor (default is 1)	<FLOAT> positive
yScale	Y-axis scale factor (default is 1)	<FLOAT> positive

Table 4–2 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.	RGB (default), RBG, GRB, GBR, BRG, BGR
inputChannels	For multiband images, specify either one (grayscale) or three integers indicating which channels to assign to red (first), green (second), and blue (third). Note that this parameter affects the source image, not the destination.	INTEGER or INTEGER INTEGER INTEGER
interleave	Controls band layout within the image: Band Interleaved by Pixel Band Interleaved by Line Band Sequential	BIP (default), BIL, BSQ
pixelOrder	If NORMAL, then the leftmost pixel appears first in the image.	NORMAL (default), REVERSE
scanlineOrder	If NORMAL, then the top scanline appears first in the image.	NORMAL (default), INVERSE

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

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

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

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

Pragmas

none

Exception

If data is not local or the source.localData attribute is not initialized, calling this method raises a DATA_NOT_LOCAL exception.

Example

Change the file format of image1 to GIF:

```
image1.process('fileFormat=GIFF');
```

Change image1 to use lower quality JPEG compression and double the length of the image along the X-axis:

```
image1.process('compressionFormat=JPEG, compressionQuality=MAXCOMPRATIO,
xScale="2.0"');
```

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

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

```
image1.process(maxScale="32 32");
```

Convert the image to TIFF:

```
DECLARE
    Image ORDSYS.ORDImage;
BEGIN
```

```
    SELECT photo INTO Image FROM emp
        WHERE ename = 'John Doe' FOR UPDATE;
        Image.process('fileFormat=TIFF');
    UPDATE emp SET photo = Image WHERE ename = 'John Doe';
END;
/
```

processCopy( ) Method

Format

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

Description

Copies an image stored internally or externally to another image stored internally in a BLOB.

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

See Table 4–1, "Image Processing Operators" and Table 4–2, "Additional Image Processing Operators for Raw Pixel and Foreign Images".

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

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

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

See Appendix C for more information on processCopy operators.

Pragmas

none

Exception

If the value of dest is NULL, calling this method raises a NULL_DESTINATION exception.

If the dest.source.isLocal attribute value is FALSE, (the destination image must be local), calling this method raises a DATA_NOT_LOCAL exception.

If the dest.source.localData attribute value is NULL (destination image must be initialized), then calling this method raises a NULL_LOCAL_DATA exception.

If the source.isLocal attribute value is 1 and the source.localData attribute value is NULL, calling this method raises a NULL_LOCAL_DATA exception.

Example

Copy an image, changing the file format, compression format, and data format in the destination image:

```
DECLARE
    Image_1 ORDSYS.ORDImage;
    Image_2 ORDSYS.ORDImage;
    mycommand VARCHAR2(400);
BEGIN
    SELECT photo, large_photo
        INTO Image_2, Image_1
        FROM emp
        WHERE ename = 'John Doe' FOR UPDATE;
    mycommand := 'fileFormat=tiff compressionFormat=packbits
                contentFormat = 8bitlut';
    Image_1.processCopy(mycommand, Image_2);
    UPDATE emp SET photo = Image_2 WHERE ename = 'John Doe';
END;
/
```

4.2.4 ORDImage Methods Associated with Properties Set and Check Operations

This section presents reference information on the ORDImage methods associated with the properties set and check operations.

setProperties Method

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 in a BLOB, or externally in a BFILE or URL. If the data is stored externally in anything other than a BFILE, the data is read into a temporary BLOB in order to determine the image characteristics.

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

Parameters

none

Usage

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

This method sets the following information about an image:

- Height in pixels
- Width in pixels
- Data size of the on-disk image in bytes
- File type (TIFF, JFIF, and so forth)
- Image type (monochrome, 8-bit grayscale, and so forth)
- Compression type (JPEG, LZW, and so forth)
- MIME type (generated based on file format)

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

Pragmas

none

Exception

If the source.isLocal attribute value is 1 and the source.localData attribute value is NULL, calling this method raises a NULL_LOCAL_DATA exception.

Example

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

```
DECLARE
    Image ORDSYS.ORDImage;
BEGIN
    INSERT INTO emp VALUES ('John Doe', 24000, 'Technical Writer', 123,
      ORDSYS.ORDImage(ORDSYS.ORDSource(empty_blob(),'file','ORDIMGDIR',
                                       'jdoe.gif',SYSDATE,0),
                      NULL,NULL,NULL,NULL,NULL,NULL,NULL));
    -- select the newly inserted row for update
    SELECT large_photo INTO Image FROM emp
           WHERE ename = 'John Doe' 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 emp SET large_photo = Image WHERE ename = 'John Doe';
 END;
/
```

Example output:

```
image width = 360
image height = 490
image size = 66318
image file type = JFIF
image type = 24BITRGB
image compression = JPEG
image mime type = image/jpeg
```

setProperties( ) Method for Foreign Images

Format

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

Description

Allows you to write the characteristics of a foreign image into the appropriate attribute fields.

> **Note:** This method will not set the MIME type attribute. You must invoke the setMimeType( ) method to set the MIME type for foreign images.

Parameters

description
Specifies the image characteristics to set for the foreign image.

Usage

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

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 D, foreign (or headerless) images either do not contain information on how to interpret the bits in the file or, *inter*Media image does not understand the information. In this case, you must set the information explicitly.

You can set the following image characteristics for foreign images, as shown in Table 4–3.

Table 4–3 **Image Characteristics for Foreign (Headerless) Files**

Field	Data Type	Description
CompressionFormat	STRING	Value must be CCITG3, CCITG4, or NONE (default).
DataOffset	INTEGER	The offset allows the image to have a header that *inter*Media image does not try to interpret. Set the offset to avoid any potential header. The value must be a positive integer less than the LOB length. Default is zero.
DefaultChannelSelection	INTEGER	For multiband images, specify either one (grayscale) or three integers indicating which channels to assign to red (first), green (second), and blue (third).
Height	INTEGER	Height of the image in pixels. Value must be a positive integer. There is no default, thus a value must be specified.
Interleave	STRING	Band layout within the image. Valid styles are: ■ BIP (default) Band Interleaved by Pixel ■ BIL Band Interleaved by Line ■ BSQ Band Sequential
NumberOfBands	INTEGER	Value must be a positive integer less than 255 describing the number of color bands in the image. Default is 3.
PixelOrder	STRING	If NORMAL (default), the leftmost pixel appears first in the file. If REVERSE, the rightmost pixel appears first.
ScanlineOrder	STRING	If NORMAL (default), the top scanline appears first in the file. If INVERSE, then the bottom scanline appears first.
UserString	STRING	A 4-character descriptive string. If used, the string is stored in the fileFormat field, appended to the file format ("OTHER:"). Default is blank and fileFormat is set to "OTHER".
Width	INTEGER	Width of the image in pixels. Value must be a positive integer. There is no default, thus a value must be specified.

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

Pragmas

none

Exception

If the description attribute value is NULL, calling this method raises a NULL_PROPERTIES_DESCRIPTION exception.

Example

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

```
DECLARE
    Image ORDSYS.ORDImage;
BEGIN
    SELECT large_photo INTO Image FROM emp
           WHERE ename = 'John Doe' FOR UPDATE;
    -- set property attributes for the image data
    Image.setProperties('width=123 height=321 compressionFormat=NONE' ||
                        ' userString=DJM dataOffset=128' ||
                        ' scanlineOrder=INVERSE pixelOrder=REVERSE' ||
                        ' interleaving=BIL numberOfBands=1' ||
                        ' defaultChannelSelection=1');
    UPDATE emp SET large_photo = Image WHERE ename = 'John Doe';
END;
/
```

checkProperties Method

Format

```
checkProperties RETURN BOOLEAN;
```

Description

Verifies that the properties stored in attributes of the image object match the properties of the image. This method should not be used for foreign images.

Parameters

none

Usage

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

Pragmas

none

Exception

none

Example

Check the image attributes:

```
DECLARE
    Image ORDSYS.ORDImage;
    properties_match BOOLEAN;
BEGIN
    SELECT large_photo INTO Image FROM emp
        WHERE ename = 'John Doe' FOR UPDATE;
    -- check that properties match the image
    properties_match := Image.checkProperties;
    IF properties_match THEN
        DBMS_OUTPUT.PUT_LINE('Check Properties succeeded');
    END IF;
END;
/
```

4.2.5 ORDImage Methods Associated with Image Attributes

This section presents reference information on the ORDImage methods associated with the image attributes.

getHeight Method

Format

```
getHeight RETURN INTEGER;
```

Description

Returns the height of an image in pixels. This method does not actually read the image; it is a simple accessor method that returns the value of the height attribute.

Parameters

none

Usage

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

Example

Get the height of an image:

```
DECLARE
    Image ORDSYS.ORDImage;
    Height INTEGER;
BEGIN
    SELECT large_photo INTO Image FROM emp
      WHERE ename = 'John Doe';
    -- get the image height
    Height := Image.getHeight;
END;
/
```

getWidth Method

Format

```
getWidth RETURN INTEGER;
```

Description

Returns the width of an image in pixels. This method does not actually read the image; it is a simple accessor method that returns the value of the width attribute.

Parameters

none

Usage

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)

Exception

none

Example

Get the width of an image:

```
DECLARE
    Image ORDSYS.ORDImage;
    Width INTEGER;
BEGIN
    SELECT large_photo INTO Image FROM emp
      WHERE ename = 'John Doe';
    -- get the image width
    Width := Image.getWidth;
END;
/
```

getContentLength Method

Format

```
getContentLength RETURN INTEGER;
```

Description

Returns the length of the image data content stored in the source. This method does not actually read the image; it is a simple access method that returns the value of the content length attribute.

Parameters

none

Usage

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)

Exception

none

Example

Get the length of the image data content stored in the source:

```
DECLARE
    Image ORDSYS.ORDImage;
    ContentLength INTEGER;
BEGIN
    SELECT large_photo INTO Image FROM emp
      WHERE ename = 'John Doe';
    -- get the image size
    ContentLength := Image.getContentLength;
END;
```

getFileFormat Method

Format

```
getFileFormat RETURN VARCHAR2;
```

Description

Returns the file type of an image (such as TIFF or JFIF). This method does not actually read the image; it is a simple accessor method that returns the value of the fileFormat attribute.

Parameters

none

Usage

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)

Exception

none

Example

Get the file type of an image:

```
DECLARE
    Image ORDSYS.ORDImage;
    FileFormat VARCHAR2(4000);
BEGIN
    SELECT large_photo INTO Image FROM emp
      WHERE ename = 'John Doe';
    -- get the image file format
    FileFormat := Image.getFileFormat;
END;
```

getContentFormat Method

Format

```
getContentFormat RETURN VARCHAR2;
```

Description

Returns the content type of an image (such as monochrome or 8-bit grayscale). This method does not actually read the image; it is a simple accessor method that returns the value of the contentFormat attribute.

Parameters

none

Usage

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)

Exception

none

Example

Get the type of an image:

```
DECLARE
    Image ORDSYS.ORDImage;
    ContentFormat VARCHAR2(4000);
BEGIN
    SELECT large_photo INTO Image FROM emp
      WHERE ename = 'John Doe';
    -- get the image content format
    ContentFormat := Image.getContentFormat;
END;
```

getCompressionFormat Method

Format

```
getCompressionFormat RETURN VARCHAR2;
```

Description

Returns the compression type of an image. This method does not actually read the image, it is a simple accessor method that returns the value of the compressionFormat attribute.

Parameters

none

Usage

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)

Exception

none

Example

Get the compression type of an image:

```
DECLARE
    Image ORDSYS.ORDImage;
    CompressionFormat VARCHAR2(4000);
BEGIN
    SELECT large_photo INTO Image FROM emp
      WHERE ename = 'John Doe';
    -- get the image compression format
    CompressionFormat := Image.getCompressionFormat;
END;
```

4.2.6 ORDImage Methods Associated with the local Attribute

This section presents reference information on the ORDImage methods associated with the local attribute.

setLocal Method

Format

```
setLocal;
```

Description

Sets the local attribute to indicate that the data is stored internally in a BLOB. When local is set, image methods look for image data in the source.localData attribute.

Parameters

none

Usage

Sets the local attribute to 1, meaning the data is stored locally in the localData attribute.

Pragmas

none

Exception

If the source.localData attribute value is NULL, calling this method raises a NULL_LOCAL_DATA exception.

Example

Set the flag to local for the data:

```
DECLARE
    Image ORDSYS.ORDImage;
BEGIN
    SELECT large_photo INTO Image FROM emp WHERE ename = 'John Doe' FOR UPDATE;
    -- set local so we look for the image in the database
    Image.setLocal;
    UPDATE emp SET large_photo = Image WHERE ename = 'John Doe';
END;
/
```

clearLocal Method

Format

```
clearLocal;
```

Description

Resets the local flag to indicate that the data is stored externally. When the local flag is set to clear, image methods look for image data using the srcLocation, srcName, and srcType attributes.

Parameters

none

Usage

This method sets the local attribute to a 0, meaning the data is stored externally or outside of Oracle8*i*.

Pragmas

none

Exception

none

Example

Clear the value of the local flag for the data:

```
DECLARE
    Image ORDSYS.ORDImage;
BEGIN
    SELECT large_photo INTO Image FROM emp WHERE ename = 'John Doe' FOR UPDATE;
    -- clear local so we look for image externally
    Image.clearLocal;
    UPDATE emp SET large_photo = Image WHERE ename = 'John Doe';
END;
/
```

isLocal Method

Format

```
isLocal RETURN BOOLEAN;
```

Description

Returns TRUE if the data is stored locally in a BLOB or FALSE if the data is stored externally.

Parameters

none

Usage

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)

Exception

none

Example

Determine whether or not the data is local:

```
DECLARE
    Image ORDSYS.ORDImage;
BEGIN
    SELECT large_photo INTO Image FROM emp WHERE ename = 'John Doe';
    -- check to see if image is stored locally
    IF Image.isLocal THEN
        DBMS_OUTPUT.PUT_LINE('Image is stored locally');
    ELSE
        DBMS_OUTPUT.PUT_LINE('Image is stored externally');
    END IF;
END;
/
```

4.2.7 ORDImage Methods Associated with the date Attribute

This section presents reference information on the ORDImage methods associated with the date attribute.

getUpdateTime Method

Format

```
getUpdateTime RETURN DATE;
```

Description

Returns the time when the object was last updated.

Parameters

none

Usage

none

Pragmas

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

Exception

none

Example

Get the updated time of some image object:

```
DECLARE
    Image ORDSYS.ORDImage;
    UpdateTime DATE;
BEGIN
    SELECT large_photo INTO Image FROM emp
      WHERE ename = 'John Doe';
    -- get the image update time
    UpdateTime := Image.getUpdateTime;
END;
/
```

setUpdateTime( ) Method

Format

```
setUpdateTime(current_time DATE);
```

Description

Sets the time when the image data was last updated. Use this method whenever you modify the image data. The methods copy( ), process( ), processCopy( ), setProperties, setMimeType( ), and export( ) call this method implicitly.

Parameters

current_time
The timestamp to be stored. Default is SYSDATE.

Usage

You must invoke this method any time you modify the image data yourself.

Pragmas

none

Exception

none

Example

Set the updated time of some image data:

```
DECLARE
    Image ORDSYS.ORDImage;
BEGIN
    SELECT large_photo INTO Image FROM emp
      WHERE ename = 'John Doe';
    -- set the image update time
    Image.setUpdateTime(SYSDATE);
END;
/
```

4.2.8 ORDImage Methods Associated with the mimeType Attribute

This section presents reference information on the ORDImage methods associated with the mimeType attribute.

getMimeType Method

Format

```
getMimeType RETURN VARCHAR2;
```

Description

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

Parameters

none

Usage

Use this method rather than accessing the mimeType attribute directly to protect yourself from potential changes to the internal representation of the ORDImage object. If the source is a file or BLOB, the MIME type information is generated.

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

Pragmas

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

Exception

none

Example

Return the MIME type:

```
DECLARE
    Image ORDSYS.ORDImage;
    MimeType VARCHAR2(4000);
BEGIN
    SELECT large_photo INTO Image FROM emp
      WHERE ename = 'John Doe';
    -- get the image mime type
    MimeType := Image.getMimeType;
```

```
END;
/
```

setMimeType( ) Method

Format

```
setMimeType(mime IN VARCHAR2);
```

Description

Allows you to set the MIME type of the image data.

Parameters

mime
The MIME type.

Usage

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

You must call this method to set the MIME type for foreign images.

Calling this method implicitly calls the setUpdateTime( ) method.

The methods setProperties, process( ), and processCopy( ) call this method implicitly.

The MIME type is extracted from the HTTP header on import for HTTP sources.

Pragmas

none

Exception

none

Example

Set the MIME type for some stored image data:

```
DECLARE
    Image ORDSYS.ORDImage;
BEGIN
    SELECT large_photo INTO Image FROM emp
```

```
      WHERE ename = 'John Doe';
    -- set the image mime type
    Image.setMimeType('image/bmp');
END;
```

4.2.9 ORDImage Methods Associated with the source Attribute

This section presents reference information on the ORDImage methods associated with the source attribute.

getContent Method

Format

```
getContent RETURN BLOB;
```

Description

Returns a handle to the local LOB storage, that is the BLOB within the ORDImage object.

Parameters

none

Usage

none

Pragmas

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

Exception

none

Example

A client accesses image data:

```
DECLARE
    Image ORDSYS.ORDImage;
    localData BLOB;
BEGIN
    SELECT large_photo INTO Image FROM emp
      WHERE ename = 'John Doe';
    -- get the image BLOB
    localData := Image.getContent;
END;
/
```

getBFILE Method

Format

```
getBFILE RETURN BFILE;
```

Description

Returns the LOB locator of the BFILE containing the image.

Parameters

none

Usage

This method constructs and returns a BFILE using the stored source.srcLocation and source.srcName attribute information. The source.srcLocation attribute must contain a defined directory object. The source.srcName attribute must be a valid file name.

Pragmas

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

Exception

If the source.srcType attribute value is NULL, calling this method raises an INCOMPLETE_SOURCE_INFORMATION exception.

If the value of srcType is other than FILE, then calling this method raises an INVALID_SOURCE_TYPE exception.

Example

```
DECLARE
    Image ORDSYS.ORDImage;
    imagebfile BFILE;
BEGIN
    SELECT large_photo INTO Image FROM emp
      WHERE ename = 'John Doe';
    -- get the image BFILE
    imagebfile := Image.getBFILE;
END;
```

deleteContent Method

Format

```
deleteContent;
```

Description

Deletes the local data from the current local source (localData).

Parameters

none

Usage

This method can be called after you export the data from the local source to an external image 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

Exception

none

Example

Delete the local data from the current local source:

```
DECLARE
    Image ORDSYS.ORDImage;
BEGIN
    SELECT large_photo INTO Image FROM emp
      WHERE ename = 'John Doe' FOR UPDATE;
    -- delete the local content of the image
    Image.deleteContent;
    UPDATE emp SET large_photo = Image WHERE ename = 'John Doe';
END;
/
```

setSource( ) Method

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 image data.

Parameters

source_type
The source type of the external data. See the "ORDSource Object Type" definition in Chapter 6 for more information.

source_location
The source location of the external data. See the "ORDSource Object Type" definition in Chapter 6 for more information.

source_name
The source name of the external data. See the "ORDSource Object Type" definition in Chapter 6 for more information.

Usage

Users can use this method to set the image data source to a new BFILE or URL. Calling this method implicitly calls the setUpdateTime( ) method.

Pragmas

none

Exception

none

Example

Set the source of the image data:

```
DECLARE
    Image ORDSYS.ORDImage;
BEGIN
    SELECT large_photo INTO Image FROM emp
      WHERE ename = 'John Doe';
    -- set source information for the image
    Image.setSource('file',
                    'ORDIMGDIR',
                    'jdoe.gif');
    UPDATE emp SET large_photo = Image WHERE ename = 'John Doe';
END;
/
```

getSource Method

Format

```
getSource RETURN VARCHAR2;
```

Description

Returns information about the external location of the image data in URL format.

Parameters

none

Usage

Possible return values are:

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

Pragmas

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

Exception

none

Example

Get the source of the image data:

```
DECLARE
    Image ORDSYS.ORDImage;
    SourceInfo VARCHAR2(4000);
BEGIN
    SELECT large_photo INTO Image FROM emp
      WHERE ename = 'John Doe';
    -- get the image source information
    SourceInfo := Image.getSource;
END;
```

getSourceType Method

Format

```
getSourceType RETURN VARCHAR2;
```

Description

Returns a string containing the type of the external image data source.

Parameters

none

Usage

This method returns a VARCHAR2 string containing the type of the external image data source, for example 'FILE'.

Pragmas

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

Exception

none

Example

Get the source type information about an image data source:

```
DECLARE
    Image ORDSYS.ORDImage;
    SourceType VARCHAR2(4000);
BEGIN
    SELECT large_photo INTO Image FROM emp
      WHERE ename = 'John Doe';
    -- get the image source type
    SourceType := Image.getSourceType;
END;
/
```

getSourceLocation Method

Format

```
getSourceLocation RETURN VARCHAR2;
```

Description

Returns a string containing the value of the external image data source location.

Parameters

none

Usage

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

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

Pragmas

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

Exception

If the value of srcLocation is NULL, then calling this method raises an ORDSource-Exceptions.INCOMPLETE_SOURCE_LOCATION exception.

Example

Get the source location information about an image data source:

```
DECLARE
    Image ORDSYS.ORDImage;
    SourceLocation VARCHAR2(4000);
BEGIN
    SELECT large_photo INTO Image FROM emp
      WHERE ename = 'John Doe';
    -- get the image source location
    SourceLocation := Image.getSourceLocation;
END;
```

getSourceName Method

Format

```
getSourceName RETURN VARCHAR2;
```

Description

Returns a string containing the name of the external image data source.

Parameters

none

Usage

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)

Exception

If the value of srcName is NULL, then calling this method raises an ORDSourceExceptions.INCOMPLETE_SOURCE_NAME exception.

Example

Get the source name information about an image data source:

```
DECLARE
    Image ORDSYS.ORDImage;
    SourceName VARCHAR2(4000);
BEGIN
    SELECT large_photo INTO Image FROM emp
      WHERE ename = 'John Doe';
    -- get the image source name
    SourceName := Image.getSourceName;
END;
/
```

import( ) Method

Format

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

Description

Transfers image data from an external image data source to a local source (localData) within an Oracle database.

Parameters

ctx
The source plug-in context information. This must be allocated. You must call the source.open( ) method; see the introduction to this chapter for more information.

Usage

Use the setSource( ) method to set the external source type, location, and name prior to calling the import( ) method.

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

After importing data from an external image data source to a local source (within an 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 the setLocal methods.

If the file format of the imported image is native, the setProperties( ) method is also called.

Pragmas

none

Exception

If the value of srcType is NULL, then calling this method raises an ORDSourceExceptions.INCOMPLETE_SOURCE_INFORMATION exception.

If the value of dlob is NULL, then calling this method raises an ORDSourceExceptions.NULL_SOURCE exception.

If the import( ) method is not supported by the source plug-in being used, an ORDSourceExceptionsMETHOD_NOT_SUPPORTED exception is raised.

See Appendix H for more information about these exceptions.

Example

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

```
DECLARE
    Image ORDSYS.ORDImage;
    ctx RAW(4000) :=NULL;
BEGIN
    -- select the image to be imported
    SELECT large_photo INTO Image FROM emp
           WHERE ename = 'John Doe' FOR UPDATE;
    -- import the image into the database
    Image.import(ctx);
    -- update the image object
    UPDATE emp SET large_photo = Image WHERE ename = 'John Doe';
END;
/
```

importFrom( ) Method

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 a local source (localData) within an Oracle database.

Parameters

ctx
The source plug-in context information. This must be allocated. You must call the source.open( ) method; see the introduction to this chapter for more information.

source_type
The source type of the image data.

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

source_name
The name of the image data.

Usage

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

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

After importing data from an external image data source to a local source (within an 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 native, the setProperties( ) method is also called.

Pragmas

none

Exception

If the value of dlob is NULL, then calling this method raises an ORDSourceExceptions.NULL_SOURCE exception.

If the importFrom( ) method is not supported by the source plug-in being used, an ORDSourceExceptions.METHOD_NOT_SUPPORTED exception is raised.

Calling this method within a source plug-in when any other exception is raised, raises an ORDSourceExceptions.SOURCE_PLUGIN_EXCEPTION exception.

See Appendix H for more information about these exceptions.

Example

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

```
DECLARE
    Image ORDSYS.ORDImage;
    ctx RAW(4000) :=NULL;
BEGIN
    -- select the image to be imported
    SELECT large_photo INTO Image FROM emp
           WHERE ename = 'John Doe' FOR UPDATE;
    -- import the image into the database
    Image.importFrom(ctx,
                     'file',
                     'ORDIMGDIR',
                     'jdoe.gif');
    -- update the image object
    UPDATE emp SET large_photo = Image WHERE ename = 'John Doe';
END;
/
```

export( ) Method

Format

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

Description

Transfers image data from a local source (localData) within an Oracle database to an external image data source.

Parameters

ctx
The source plug-in context information.

source_type
The source type of the location to where the data is to be exported.

source_location
The location to where the image data is to be exported.

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

Usage

After exporting image data, all image attributes remain unchanged. Source attributes: source type, source location, and source name information are updated. After calling the export method, you can call the deleteContent method to delete the content of the local data.

There is no server-side native support for the export method; this method is available for user-defined sources that can support the export method.

Pragmas

none

Exception

If the value of srcType is NULL, then calling this method raises an ORDSourceExceptions.INCOMPLETE_SOURCE_INFORMATION exception.

If the export( ) method is not supported by the source plug-in being used, an ORDSourceExceptions.METHOD_NOT_SUPPORTED exception is raised.

Calling this method within a source plug-in when any other exception is raised, raises an ORDSourceExceptions.SOURCE_PLUGIN_EXCEPTION exception.

See Appendix H for more information about these exceptions.

Example

none

4.2.10 ORDImage Methods Associated with Image Migration

This section presents reference information on the ORDImage methods associated with image migration relative to converting old ORDImgB images and ORDImgF images to new ORDImage images.

migrateFromORDImgB( ) Method

Format

```
migrateFromORDImgB(old_object ORDImgB);
```

Description

Allows you to migrate old ORDImgB images to the new ORDImage object.

Parameters

old_object
The old ORDImgB image.

Usage

This method copies from the source BLOB to the destination BLOB, copies all the image attributes from the old object to the new object, and sets the update time and local attribute.

Pragmas

none

Exception

If the value of src (old_object) is NULL, calling this method raises a NULL_SOURCE exception.

If the value of dest is NULL (ORDImage), calling this method raises a NULL_DESTINATION exception.

If the value of src.content is NULL (old.object content attribute), calling this method raises a NULL_CONTENT exception.

If the dest.source.localData value is NULL (dest ORDImage source.localData), calling this method raises a NULL_LOCAL_DATA exception.

Example

Migrate an old ORDImgB image to a new ORDImage image:

```
DECLARE
```

```
    Image ORDSYS.ORDImage;
    old_image ORDSYS.ORDIMGB;
BEGIN
    -- Select the old image
    SELECT imageb INTO old_image FROM old_images WHERE id = 1;
    -- Select the new image
    SELECT large_photo INTO Image FROM emp WHERE ename = 'John Doe' FOR UPDATE;
    -- Migrate from the old to the new
    Image.migrateFromORDImgB(old_image);
    UPDATE emp SET large_photo = Image WHERE ename = 'John Doe';
END;
/
```

migrateFromORDImgF( ) Method

Format

```
migrateFromORDImgF(old_object ORDImgF);
```

Description

Allows you to migrate old ORDImgF images to the new ORDImage object.

Parameters

old_object
The old ORDImgF image.

Usage

This method extracts the directory name and file name from the source and copies them to the srcLocation and srcName attributes of the destination. It also copies all image attributes from the old image object to the new image object, sets the updateTime attribute, and clears the local attribute.

Pragmas

none

Exception

none

Example

Migrate an old ORDImgF image to a new ORDImage image:

```
DECLARE
    Image ORDSYS.ORDImage;
    old_image ORDSYS.ORDIMGF;
BEGIN
    -- Select the old image
    SELECT imagef INTO old_image FROM old_images WHERE id = 1;
    -- Select the new image
    SELECT large_photo INTO Image FROM emp WHERE ename = 'John Doe' FOR UPDATE;
    -- Migrate from the old to the new
```

```
    Image.migrateFromORDImgf(old_image);
    UPDATE emp SET large_photo = Image WHERE ename = 'John Doe';
END;
/
```

5

ORDVideo Reference Information

Oracle8*i* *inter*Media contains the following information about the ORDVideo type:

- Object type -- see Section 5.1.
- Methods -- see Section 5.2.
- Packages or PL/SQL plug-ins -- see Section 5.3.

The examples in this chapter assume that the test video table TVID has been created and filled with data. This table was created using the SQL statements described in Section 5.2.1.

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

Methods invoked at the ORDSource level that are handed off to the source plug-in for processing have ctx (RAW(4000)) 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 should invoke the closeSource( ) method.

Methods invoked from a source plug-in call have the first argument as ctx (RAW(4000)).

Methods invoked at the ORDVideo level that are handed off to the format plug-in for processing have ctx (RAW(4000)) 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, not all source or format plug-ins will use the ctx argument, but if you code as previously described, your application should work with any current or future source or format plug-in.

5.1 Object Types

Oracle8*i* *inter*Media describes the ORDVideo object type, which supports the storage and management of video data.

ORDVideo Object Type

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

```
CREATE OR REPLACE TYPE ORDVideo
AS OBJECT
(
  -- ATTRIBUTES
description         VARCHAR2(4000),
source              ORDSource,
format              VARCHAR2(31),
mimeType            VARCHAR2(4000),
comments            CLOB,
  -- VIDEO RELATED ATTRIBUTES
width               INTEGER,
height              INTEGER,
frameResolution     INTEGER,
frameRate           INTEGER,
videoDuration       INTEGER,
numberOfFrames      INTEGER,
compressionType     VARCHAR2(4000),
numberOfColors      INTEGER,
bitRate             INTEGER,

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

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

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

```
                                    cmd       IN VARCHAR2,
                                    arguments IN VARCHAR2,
                                    result    OUT RAW)
                RETURN RAW,

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

MEMBER PROCEDURE  setLocal,
MEMBER PROCEDURE  clearLocal,

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

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

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

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

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

MEMBER FUNCTION getContentLength(ctx IN OUT RAW) RETURN INTEGER,
PRAGMA RESTRICT_REFERENCES(getContentLength, WNDS, WNPS, RNDS, RNPS),

MEMBER FUNCTION getContentInLob(
                                ctx      IN OUT RAW,
                                dest_lob IN OUT NOCOPY BLOB,
```

```
                                mimeType OUT VARCHAR2,
                                format   OUT VARCHAR2),

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

MEMBER PROCEDURE deleteContent,

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

-- Methods associated with the comments attribute
MEMBER PROCEDURE appendToComments(amount IN BINARY_INTEGER,
                                  buffer IN VARCHAR2),
MEMBER PROCEDURE writeToComments(offset IN INTEGER,
                                 amount IN BINARY_INTEGER,
                                 buffer IN VARCHAR2),
MEMBER FUNCTION readFromComments(offset IN INTEGER,
                                 amount IN BINARY_INTEGER := 32767)
               RETURN VARCHAR2,
PRAGMA RESTRICT_REFERENCES(readFromComments, WNDS, WNPS, RNDS, RNPS),

MEMBER FUNCTION locateInComments(pattern    IN VARCHAR2,
                                 offset     IN INTEGER := 1,
                                 occurrence IN INTEGER := 1)
                RETURN INTEGER,
MEMBER PROCEDURE trimComments(newlen IN INTEGER),
MEMBER PROCEDURE eraseFromComments(amount IN OUT NOCOPY INTEGER,
                                   offset IN INTEGER := 1),
MEMBER PROCEDURE deleteComments,
MEMBER PROCEDURE loadCommentsFromFile(fileobj  IN BFILE,
                                      amount   IN INTEGER,
```

```
                                    from_loc IN INTEGER :=1,
                                    to_loc   IN INTEGER :=1),
MEMBER PROCEDURE copyCommentsOut(dest     IN OUT NOCOPY CLOB,
                                 amount   IN INTEGER,
                                 from_loc IN INTEGER :=1,
                                 to_loc   IN INTEGER :=1),
MEMBER FUNCTION compareComments(
                  compare_with_lob        IN CLOB,
                  amount                  IN INTEGER := 4294967295,
                  starting_pos_in_comment IN INTEGER := 1,
                  starting_pos_in_compare IN INTEGER := 1)
               RETURN INTEGER,
PRAGMA RESTRICT_REFERENCES(compareComments, WNDS, WNPS, RNDS, RNPS),

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

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

MEMBER PROCEDURE setFrameSize(knownWidth IN INTEGER, knownHeight IN INTEGER),
MEMBER PROCEDURE getFrameSize(retWidth OUT INTEGER, retHeight OUT INTEGER),
PRAGMA RESTRICT_REFERENCES(getFrameSize, WNDS, WNPS, RNDS, RNPS),
MEMBER PROCEDURE getFrameSize(
                           ctx IN OUT RAW,
                           retWidth OUT INTEGER,
                           retHeight OUT INTEGER),

MEMBER PROCEDURE setFrameResolution(knownFrameResolution IN INTEGER),
MEMBER FUNCTION getFrameResolution RETURN INTEGER,
PRAGMA RESTRICT_REFERENCES(getFrameResolution, WNDS, WNPS, RNDS, RNPS),
MEMBER FUNCTION getFrameResolution(ctx IN OUT RAW) RETURN INTEGER,

MEMBER PROCEDURE setFrameRate(knownFrameRate IN INTEGER),
MEMBER FUNCTION getFrameRate RETURN INTEGER,
PRAGMA RESTRICT_REFERENCES(getFrameRate, WNDS, WNPS, RNDS, RNPS),
MEMBER FUNCTION getFrameRate(ctx IN OUT RAW) RETURN INTEGER,

MEMBER PROCEDURE setVideoDuration(knownVideoDuration IN INTEGER),
MEMBER FUNCTION getVideoDuration RETURN INTEGER,
PRAGMA RESTRICT_REFERENCES(getVideoDuration, WNDS, WNPS, RNDS, RNPS),
MEMBER FUNCTION getVideoDuration(ctx IN OUT RAW) RETURN INTEGER,
```

```
MEMBER PROCEDURE setNumberOfFrames(knownNumberOfFrames IN INTEGER),
MEMBER FUNCTION getNumberOfFrames RETURN INTEGER,
PRAGMA RESTRICT_REFERENCES(getNumberOfFrames, WNDS, WNPS, RNDS, RNPS),
MEMBER FUNCTION getNumberOfFrames(ctx IN OUT RAW) RETURN INTEGER,

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

MEMBER PROCEDURE setNumberOfColors(knownNumberOfColors IN INTEGER),
MEMBER FUNCTION getNumberOfColors RETURN INTEGER,
PRAGMA RESTRICT_REFERENCES(getNumberOfColors, WNDS, WNPS, RNDS, RNPS),
MEMBER FUNCTION getNumberOfColors(ctx IN OUT RAW) RETURN INTEGER,

MEMBER PROCEDURE setBitRate(knownBitRate IN INTEGER),
MEMBER FUNCTION getBitRate RETURN INTEGER,
PRAGMA RESTRICT_REFERENCES(getBitRate, WNDS, WNPS, RNDS, RNPS),
MEMBER FUNCTION getBitRate(ctx IN OUT RAW) RETURN INTEGER,

MEMBER PROCEDURE setKnownAttributes(
                                  knownFormat IN VARCHAR2,
                                  knownWidth IN INTEGER,
                                  knownHeight IN INTEGER,
                                  knownFrameResolution IN INTEGER,
                                  knownFrameRate IN INTEGER,
                                  knownVideoDuration IN INTEGER
                                  knownNumberOfFrames IN INTEGER,
                                  knownCompressionType IN VARCHAR2,
                                  knownNumberOfColors IN INTEGER,
                                  knownBitRate IN INTEGER),

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

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

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

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

where:

- description: the description of the video object.
- source: the ORDSource where the video data is to be found.
- format: the format in which the video data is stored.
- mimeType: the MIME type information.
- comments: the comment 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.

5.2 Methods

This section presents reference information on the Oracle8*i* *inter*Media methods used for video data manipulation. These methods are described in the following groupings:

ORDVideo Methods Associated with the updateTime Attribute

- getUpdateTime: returns the time when the video object was last updated.
- setUpdateTime( ): sets the update time for the video object.

ORDVideo Methods Associated with the title Attribute

- setDescription( ): sets the description of the video data.
- getDescription: returns the description of the video data.

ORDVideo Methods Associated with mimeType Attribute

- setMimeType( ): sets the MIME type of the stored video data.
- getMimeType: returns the MIME type for video data.

ORDVideo Methods Associated with the source Attribute

- processSourceCommand( ): sends a command and related arguments to the source plug-in.
- isLocal: returns TRUE if the data is stored locally in a BLOB or FALSE if the data is external.
- setLocal: sets a flag to indicate that the data is stored locally in a BLOB.
- clearLocal: clears the flag to indicate that the data is stored externally.
- setSource( ): sets the source information to where video data is to be found.
- getSource: returns a formatted string containing complete information about the external data source formatted as a URL.
- getSourceType: returns the external source type of the video data.
- getSourceLocation: returns the external source location of the video data.
- getSourceName: returns the external source name of the video data.
- import( ): transfers data from an external data source (specified by calling set-SourceInformation( )) to the local source (localData) within an Oracle database,

setting the value of the local attribute to "1", meaning local and updating the timestamp.

- importFrom( ): transfers data from the specified external data source (source type, location, name) to the local source (localData) within an Oracle database, setting the value of the local attribute to "1", meaning local and updating the timestamp.
- export( ): transfers data from a local source (localData) within an Oracle database to the specified external data source, setting the value of the local attribute to "0", meaning external, and storing source information in the source.

> **Note:** Sources natively supported by *inter*Media are not writable and therefore do not support the export method. User-defined sources may support the export method.

- getContentLength( ): returns the length of the data source (as number of bytes).
- getContentInLob( ): returns content into a temporary LOB.
- getContent: returns the handle to the BLOB used to store contents locally.
- deleteContent: deletes the content of the local BLOB.

ORDAudio Methods Associated with File Operations

- openSource( ): opens a data source.
- closeSource( ): closes a data source.
- trimSource( ): trims a data source.
- readFromSource( ): reads a buffer of n bytes from a source beginning at a start position.
- writeToSource( ): writes a buffer of n bytes to a source beginning at a start position.

ORDVideo Methods Associated with the comments Attribute

- appendToComments( ): appends a specified buffer and amount of comment data to the end of the video data comments.
- writeToComments( ): writes a specified buffer and amount of comment data to the video data comments beginning at the specified offset.

- readFromComments( ): reads a specified amount of comment data from the video data comments beginning at the specified offset.
- locateInComments( ): matches and locates the occurrence of the specified pattern of characters in the video data comments.
- trimComments( ): trims the video data comments to the specified length.
- eraseFromComments( ): erases the specified amount of comment data from the video data comments beginning at the specified offset.
- deleteComments: deletes the video data comments.
- loadCommentsFromFile( ): loads the comments from the specified BFILE into the video data comments.
- copyCommentsOut( ): copies the video data comments to the given Character LOB (CLOB).
- compareComments( ): compares video data comments with the comments of another specified CLOB of video data.
- getCommentLength: returns the length of the video data comments.

ORDVideo Methods Associated with Video Attributes Accessors

The following methods are supported only by user-defined format plug-ins:

- setFormat( ): sets the object attribute value of the format of the video data.
- getFormat: returns the object attribute value of the format in which the video data is stored.
- getFormat( ): calls the format plug-in to read the actual format embedded in the stored video data.
- setFrameSize( ): sets the object attribute value of the width and height in pixels of each frame in the video data.
- getFrameSize: returns the object attribute value of the width and height in pixels of each frame in the video data.
- getFrameSize( ): calls the format plug-in to read the actual width and height in pixels of each frame embedded in the stored audio data.
- setFrameResolution( ): sets the object attribute value of the number of pixels per inch of frames in the video data.
- getFrameResolution: returns the object attribute value of the number of pixels per inch of frames in the video data.

- getFrameResolution( ): calls the format plug-in to read the actual frame resolution embedded in the stored video data.
- setFrameRate( ): sets the object attribute value of the rate in frames per second at which the video data was recorded.
- getFrameRate: returns the object attribute value of the rate in frames per second at which the video data was recorded.
- getFrameRate( ): calls the format plug-in to read the actual frame rate embedded in the stored video data.
- setVideoDuration( ): sets the object attribute value of the total time it takes to play the entire video data.
- getVideoDuration: returns the object attribute value of the total time it takes to play the entire video data.
- getVideoDuration( ): calls the format plug-in to read the actual video duration embedded in the stored video data.
- setNumberOfFrames( ): sets the object attribute value of the total number of frames in the video data.
- getNumberOfFrames: returns the object attribute value of the total number of frames in the video data.
- getNumberOfFrames( ): calls the format plug-in to read the actual number of frames embedded in the stored video data.
- setNumberOfColors( ): sets the object attribute value of the number of colors in the video data.
- getNumberOfColors: returns the object attribute value of the number of colors in the video data.
- getNumberOfColors( ): calls the format plug-in to read the actual number of colors embedded in the stored video data.
- setBitRate( ): sets the object attribute value of the bit rate in the video data.
- getBitRate: returns the object attribute value of the bit rate in the video data.
- getBitRate( ): calls the format plug-in to read the actual bit rate embedded in the stored video data.
- setKnownAttributes( ): sets known video attributes including format, frame size, frame resolution, frame rate, video duration, number of frames, compres-

sion type, number of colors, and bit rate of the video data. The parameters are passed in with this call.

- setProperties( ): reads the video data to get the values of the object attributes and then stores them in the object. For the known attributes that ORDVideo understands, it sets the properties for these attributes, which include: format, frame size, frame resolution, frame rate, video duration, number of frames, compression type, number of colors, and bit rate of the video data.
- checkProperties( ): calls the format plug-in to check the properties including format, frame size, frame resolution, frame rate, video duration, number of frames, compression type, number of colors, and bit rate of the video data; it returns a Boolean value TRUE if the properties stored in object attributes match those in the video data.
- getAttribute( ): returns the value of the requested attribute. This method is only available for user-defined format plug-ins.
- getAllAttributes( ): returns a formatted string for convenient client access. For natively supported formats, the string includes the following list of video data attributes separated by a comma (','): format, frameSize, frameResolution, frameRate, videoDuration, numberOfFrames, compressionType, numberOfColors, and bitRate. The string is defined by the user-defined format plug-in.

ORDVideo Methods Associated with Processing Video Data

- processVideoCommand( ): sends commands and related arguments to the format plug-in for processing.

For more information on object types and methods, see the *Oracle8i Concepts* manual.

5.2.1 Example Table Definitions

The methods described in this reference chapter show examples based on a test video table TVID. Refer to the TVID table definition that follows when reading through the examples in Section 5.2.2 through Section 5.2.9:

TVID Table Definition

```
CREATE TABLE TVID(n NUMBER, vid ORDSYS.ORDVIDEO)
storage (initial 100K next 100K pctincrease 0);

INSERT INTO TVID VALUES(1, ORDSYS.ORDVideo(
                          NULL,
        ORDSYS.ORDSOURCE(EMPTY_BLOB(), NULL, NULL, NULL, SYSDATE, NULL),
```

```
                              NULL,
                              NULL,
                              EMPTY_CLOB(),0,0,0,0,0,0,0,0,0)
                         );
INSERT INTO TVID VALUES(2, ORDSYS.ORDVideo(
                              NULL,
                              ORDSYS.ORDSOURCE(EMPTY_BLOB(), NULL, NULL, NULL,
SYSDATE,NULL),
                              NULL,
                              NULL,
                              EMPTY_CLOB(),0,0,0,0,0,0,0,0,0)
                         );
```

5.2.2 ORDVideo Methods Associated with the updateTime Attribute

This section presents reference information on the ORDVideo methods associated with the updateTime attribute.

getUpdateTime Method

Format

```
getUpdateTime RETURN DATE;
```

Description

Returns the time when the object was last updated.

Parameters

none

Usage

none

Pragmas

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

Exception

none

Example

Get the updated time of some video data:

```
DECLARE
 obj ORDSYS.ORDVideo;
BEGIN
 SELECT TVID INTO obj FROM TVID WHERE N = 1 ;
 DBMS_OUTPUT.PUT_LINE(TO_CHAR(obj.getUpdateTime,'MM-DD-YYYY HH24:MI:SS'));
END;
/
```

setUpdateTime( ) Method

Format

```
setUpdateTime(current_time DATE);
```

Description

Sets the time when the video data was last updated. Use this method whenever you modify the video data. The methods setDescription( ), setMimeType( ), setSource( ), import( ), importFrom( ), export( ), deleteContent, and all set video accessors call this method implicitly.

Parameters

current_time
The timestamp to be stored. Default is SYSDATE.

Usage

You must invoke this method whenever you modify the video data.

Pragmas

none

Exception

none

Example

See also the example in the getUpdateTime Method on page 5-15.

Set the updated time of some video data:

```
DECLARE
 obj ORDSYS.ORDVideo;
BEGIN
 SELECT vid INTO obj FROM TVID WHERE N = 1;
 obj.setUpdateTime(SYSDATE);
 UPDATE TVID SET vid=obj WHERE N = 1;
 COMMIT;
END;
```

5.2.3 ORDVideo Methods Associated with the description Attribute

This section presents reference information on the ORDVideo methods associated with the description attribute.

setDescription( ) Method

Format

```
setDescription (user_description IN VARCHAR2);
```

Description

Sets the description of the video data.

Parameters

user_description
The description of the video data.

Usage

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 Oracle8*i* *inter*-Media make use of this description attribute to present video data to users.

Calling this method implicitly calls the setUpdateTime( ) method.

Pragmas

none

Exception

none

Example

Set the description attribute for some video data:

```
DECLARE
  obj ORDSYS.ORDVideo;
BEGIN
  SELECT vid INTO obj FROM TVID  WHERE N=1 FOR UPDATE;
  DBMS_OUTPUT.PUT_LINE('writing description');
  DBMS_OUTPUT.PUT_LINE('-------------');
```

```
  obj.setDescription('video1');
  DBMS_OUTPUT.PUT_LINE(obj.getDescription);
  UPDATE TVID SET vid=obj WHERE N=1;
  COMMIT;
END;
/
```

getDescription Method

Format

```
getDescription RETURN VARCHAR2;
```

Description

Returns the description of the video data.

Parameters

none

Usage

none

Pragmas

Pragma RESTRICT_REFERENCES(getTitle, WNDS, WNPS, RNDS, RNPS)

Exception

If you call the setDescription method and the description is not set, a DESCRIPTION_IS_NOT_SET exception is raised.

Example

See the example in the setDescription( ) Method on page 5-18.

5.2.4 ORDVideo Methods Associated with the mimeType Attribute

This section presents reference information on the ORDVideo methods associated with the mimeType attribute.

setMimeType( ) Method

Format

```
setMimeType(mime IN VARCHAR2);
```

Description

Allows you to set the MIME type of the video data.

Parameters

mime
The MIME type.

Usage

Calling this method implicitly calls the setUpdateTime( ) method.

Call the setMimeType( ) method to set the MIME type if the source is a file or BLOB.

The MIME type is extracted from the HTTP header on import for HTTP sources.

Pragmas

none

Exception

If you call the setMimeType( ) method and the value for MIME type is NULL, an INVALID_MIME_TYPE exception is raised.

Example

Set the MIME type for some stored video data:

```
DECLARE
  obj ORDSYS.ORDVideo;
BEGIN
  SELECT vid INTO obj FROM TVID  WHERE N=1 FOR UPDATE;
  DBMS_OUTPUT.PUT_LINE('writing mimetype');
  DBMS_OUTPUT.PUT_LINE('----------------');
  obj.setMimeType('video/avi');
  DBMS_OUTPUT.PUT_LINE(obj.getMimeType);
```

```
  UPDATE TVID SET vid=obj WHERE N=1;
  COMMIT;
END;
/
```

getMimeType Method

Format

```
getMimeType RETURN VARCHAR2;
```

Description

Returns the MIME type for the video data.

Parameters

none

Usage

If the source is an HTTP server, the MIME type information is read from the HTTP header information. If the source is a file or BLOB, you must call the setMimeType( ) method to set the MIME type.

Pragmas

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

Exception

none

Example

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

5.2.5 ORDVideo Methods Associated with the source Attribute

This section presents reference information on the ORDVideo methods associated with the source attribute.

processSourceCommand( ) Method

Format

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

Description

Allows you to 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 must be allocated. You must call the openSource( ) method; see the introduction to this chapter for more information.

cmd
Any command recognized by the source plug-in.

arguments
The arguments of the command.

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

Usage

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

Pragmas

none

Exception

If the value of srcType is NULL, then calling this method raises an ORDSourceExceptions.INCOMPLETE_SOURCE_INFORMATION exception.

If the processSourceCommand( ) method is not supported by the source plug-in being used, an ORDSourceExceptions.METHOD_NOT_SUPPORTED exception is raised.

Calling this method within a source plug-in when any other exception is raised, raises an ORDSourceExceptions.SOURCE_PLUGIN_EXCEPTION exception.

See Appendix H for more information about these exceptions.

Example

Process some commands:

```
DECLARE
  obj ORDSYS.ORDVideo;
  res RAW(4000);
  result RAW(4000);
  command VARCHAR(4000);
  argList VARCHAR(4000);
  ctx RAW(4000) :=NULL;
BEGIN
select vid into obj from TVID where N =1 for UPDATE;
-- assign command
-- assign argList
res := obj.processSourceCommand (ctx, command,
                                        argList, result);
UPDATE TVID SET vid=obj WHERE N=1 ;
COMMIT;
  EXCEPTION
        WHEN ORDSYS.ORDSourceExceptions.METHOD_NOT_SUPPORTED THEN
         DBMS_OUTPUT.put_line('Source METHOD_NOT_SUPPORTED caught');
        WHEN ORDSYS.ORDSourceExceptions.SOURCE_PLUGIN_EXCEPTION THEN
         DBMS_OUTPUT.put_line('SOURCE PLUGIN EXCEPTION caught');
        WHEN ORDSYS.ORDVideoExceptions.METHOD_NOT_SUPPORTED THEN
         DBMS_OUTPUT.put_line('VIDEO METHOD_NOT_SUPPORTED EXCEPTION caught');
        WHEN ORDSYS.ORDVideoExceptions.VIDEO_PLUGIN_EXCEPTION THEN
         DBMS_OUTPUT.put_line('VIDEO PLUGIN EXCEPTION caught');
        WHEN OTHERS THEN
         DBMS_OUTPUT.PUT_LINE('EXCEPTION caught');
END;
/
```

isLocal Method

Format

```
isLocal RETURN BOOLEAN;
```

Description

Returns TRUE if the data is stored locally in a BLOB or FALSE if the data is stored externally.

Parameters

none

Usage

If the local attribute value is set to 1 or NULL, this method returns TRUE, otherwise this method returns FALSE.

Pragmas

Pragma RESTRICT_REFERENCES(getLocal, WNDS, WNPS, RNDS, RNPS)

Exception

none

Example

Determine whether or not the data is local:

```
DECLARE
 obj ORDSYS.ORDVideo;
BEGIN
 SELECT vid INTO obj FROM TVID WHERE N = 1 FOR UPDATE;
 if(obj.isLocal) then
   DBMS_OUTPUT.put_line('local is true');
 else
   DBMS_OUTPUT.put_line('local is false');
 endif;
END;
/
```

setLocal Method

Format

```
setLocal;
```

Description

Sets the local attribute to indicate that the data is stored internally in a BLOB. When the local flag is set, video methods look for video data in the source.localData attribute.

Parameters

none

Usage

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

Pragmas

none

Exception

none

Example

Set the flag to local for the data:

```
DECLARE
 obj ORDSYS.ORDVideo;
BEGIN
 SELECT s INTO obj FROM TVID WHERE N = 1 FOR UPDATE;
 obj.setLocal;
 UPDATE TVID SET s=obj WHERE N = 1;
 COMMIT;
END;
/
```

clearLocal Method

Format

```
clearLocal;
```

Description

Resets the local flag to indicate that the data is stored externally. When the local flag is set to clear, video methods look for video data using the srcLocation, srcName, and srcType attributes.

Parameters

none

Usage

This method sets the local attribute to a 0, meaning the data is stored externally or outside of Oracle8*i*.

Pragmas

none

Exception

none

Example

Clear the value of the local flag for the data:

```
DECLARE
 obj ORDSYS.ORDVideo;
BEGIN
 SELECT s INTO obj FROM TVID WHERE N = 1 FOR UPDATE;
 obj.clearLocal;
 UPDATE TVID SET s=obj WHERE N = 1;
 COMMIT;
END;
/
```

setSource( ) Method

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 video data.

Parameters

source_type
The source type of the external data. See the "ORDSource Object Type" definition in Chapter 6 for more information.

source_location
The source location of the external data. See the "ORDSource Object Type" definition in Chapter 6 for more information.

source_name
The source name of the external data. See the "ORDSource Object Type" definition in Chapter 6 for more information.

Usage

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

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

Calling this method implicitly calls the setUpdateTime( ) method.

Pragmas

none

Exception

none

Example

Change the source to the exported file prior to exporting the video data:

```
DECLARE
  obj ORDSYS.ORDVideo;
BEGIN
  SELECT vid INTO obj FROM TVID  WHERE N=1 FOR UPDATE;
  DBMS_OUTPUT.PUT_LINE('setting and getting source');
  DBMS_OUTPUT.PUT_LINE('-------------------------');
  obj.setSource('LOCAL','VIDEODIR','video.dat');
  DBMS_OUTPUT.PUT_LINE(obj.getSource);
  UPDATE TVID SET vid=obj WHERE N=1;
  COMMIT;
END;
/
```

getSource Method

Format

```
getSource RETURN VARCHAR2;
```

Description

Returns information about the external location of the video data in URL format.

Parameters

none

Usage

Possible return values are:

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

Pragmas

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

Exception

none

Example

See the example in the setSource( ) Method on page 5-32.

getSourceType Method

Format

```
getSourceType RETURN VARCHAR2;
```

Description

Returns a string containing the type of the external video data source.

Parameters

none

Usage

This method returns a VARCHAR2 string containing the type of the external video data source, for example 'FILE'.

Pragmas

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

Exception

none

Example

Get the source type information about a video data source:

```
DECLARE
  obj ORDSYS.ORDVideo;
BEGIN
  SELECT vid INTO obj FROM TVID  WHERE N=1 FOR UPDATE;
  DBMS_OUTPUT.PUT_LINE('setting and getting source');
  DBMS_OUTPUT.PUT_LINE('-------------------------');
  -- set source to a file
  obj.setSource('FILE','VIDEODIR','MV1.AVI');
  -- get source information
  DBMS_OUTPUT.put_line(obj.getSource);
  DBMS_OUTPUT.put_line(obj.getSourceType);
  DBMS_OUTPUT.put_line(obj.getSourceLocation);
  DBMS_OUTPUT.put_line(obj.getSourceName);
```

```
  UPDATE TVID SET vid=obj WHERE N=1;
  COMMIT;
  EXCEPTION
        WHEN ORDSYS.ORDVideoExceptions.METHOD_NOT_SUPPORTED THEN
        DBMS_OUTPUT.PUT_LINE('Source not specified');
END;
/
```

getSourceLocation Method

Format

```
getSourceLocation RETURN VARCHAR2;
```

Description

Returns a string containing the value of the external video data source location.

Parameters

none

Usage

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

Pragmas

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

Exception

If the value of srcLocation is NULL, then calling this method raises an ORDSource-Exceptions.INCOMPLETE_SOURCE_LOCATION exception.

Example

See the example in the getSourceType Method on page 5-34.

getSourceName Method

Format

```
getSourceName RETURN VARCHAR2;
```

Description

Returns a string containing the name of the external video data source.

Parameters

none

Usage

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

Pragmas

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

Exception

If the value of srcName is NULL, then calling this method raises an ORDSourceExceptions.INCOMPLETE_SOURCE_NAME exception.

Example

See the example in the getSourceType Method on page 5-34.

import( ) Method

Format

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

Description

Transfers video data from an external video data source to a local source (local-Data) within an Oracle database.

Parameters

ctx
The source plug-in context information. This must be allocated. You must call the openSource( ) method; see the introduction to this chapter for more information.

Usage

Use the setSource( ) method to set the external source type, location, and name prior to calling import.

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

After importing data from an external video data source to a local source (within an 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.

Pragmas

none

Exception

If the value of srcType is NULL, then calling this method raises an ORDSourceExceptions.INCOMPLETE_SOURCE_INFORMATION exception.

If the value of dlob is NULL, then calling this method raises an ORDSourceExceptions.NULL_SOURCE exception.

If the import( ) method is not supported by the source plug-in being used, an ORDSourceExceptions.METHOD_NOT_SUPPORTED exception is raised.

Calling this method within a source plug-in when any other exception is raised, raises an ORDSourceExceptions.SOURCE_PLUGIN_EXCEPTION exception.

See Appendix H for more information about these exceptions.

Example

Import video data by first setting the source and then importing it:

```
DECLARE
  obj ORDSYS.ORDVideo;
  ctx RAW(4000) :=NULL;
BEGIN
  SELECT vid INTO obj FROM TVID  WHERE N=1 FOR UPDATE;
  DBMS_OUTPUT.PUT_LINE('setting and getting source');
  DBMS_OUTPUT.PUT_LINE('-------------------------');
  -- set source to a file
  obj.setSource('FILE','VIDEODIR','testvid.dat');
  -- get source information
  DBMS_OUTPUT.PUT_LINE(obj.getSource);
  -- import data
  obj.import(ctx);
  -- check size
  DBMS_OUTPUT.PUT_LINE(TO_CHAR(obj.getContentLength(ctx)));
  DBMS_OUTPUT.PUT_LINE(obj.getSource);
  DBMS_OUTPUT.PUT_LINE('deleting contents');
  DBMS_OUTPUT.PUT_LINE('----------------');
  obj.deleteContent;
  DBMS_OUTPUT.PUT_LINE(TO_CHAR(obj.getContentLength(ctx)));
  UPDATE TVID SET vid=obj WHERE N=1;
  COMMIT;
END;
/
```

importFrom( ) Method

Format

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

Description

Transfers video data from the specified external video data source to a local source (localData) within an Oracle database.

Parameters

ctx
The source plug-in context information. This must be allocated. You must call the openSource( ) method; see the introduction to this chapter for more information.

source_type
The source type of the video data.

source_location
The location from where the video data is to be imported.

source_name
The name of the video data.

Usage

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

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

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

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

Pragmas

none

Exception

If the value of dlob is NULL, then calling this method raises an ORDSourceExceptions.NULL_SOURCE exception.

If the importFrom( ) method is not supported by the source plug-in being used, an ORDSourceExceptions.METHOD_NOT_SUPPORTED exception is raised.

Calling this method within a source plug-in when any other exception is raised, raises an ORDSourceExceptions.SOURCE_PLUGIN_EXCEPTION exception.

See Appendix H for more information about these exceptions.

Example

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

```
DECLARE
  obj ORDSYS.ORDVideo;
  ctx RAW(4000) :=NULL;
BEGIN
  SELECT vid INTO obj FROM TVID  WHERE N=1 FOR UPDATE;
  DBMS_OUTPUT.PUT_LINE('setting and getting source');
  DBMS_OUTPUT.PUT_LINE('-------------------------');
  -- import data
  obj.importFrom(ctx,'FILE','VIDEODIR','MV1.AVI');
  -- check size
  DBMS_OUTPUT.PUT_LINE(TO_CHAR(obj.getContentLength(ctx)));
  DBMS_OUTPUT.PUT_LINE(TO_CHAR(DBMS_LOB.GETLENGTH(obj.getContent)));
  DBMS_OUTPUT.PUT_LINE(obj.getSource);
  DBMS_OUTPUT.PUT_LINE('deleting contents');
  DBMS_OUTPUT.PUT_LINE('----------------');
  obj.deleteContent;
  DBMS_OUTPUT.PUT_LINE(TO_CHAR(obj.getContentLength(ctx)));
  UPDATE TVID SET vid=obj WHERE N=1;
  COMMIT;
  EXCEPTION
        WHEN ORDSYS.ORDSourceExceptions.METHOD_NOT_SUPPORTED THEN
         DBMS_OUTPUT.put_line('Source METHOD_NOT_SUPPORTED caught');
        WHEN ORDSYS.ORDSourceExceptions.SOURCE_PLUGIN_EXCEPTION THEN
         DBMS_OUTPUT.put_line('SOURCE PLUGIN EXCEPTION caught');
        WHEN ORDSYS.ORDVideoExceptions.METHOD_NOT_SUPPORTED THEN
         DBMS_OUTPUT.put_line('VIDEO METHOD_NOT_SUPPORTED EXCEPTION caught');
```

```
        WHEN ORDSYS.ORDVideoExceptions.VIDEO_PLUGIN_EXCEPTION THEN
         DBMS_OUTPUT.put_line('VIDEO PLUGIN EXCEPTION caught');
        WHEN OTHERS THEN
         DBMS_OUTPUT.PUT_LINE('EXCEPTION Caught');
END;
/
```

export( ) Method

Format

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

Description

Transfers video data from a local source (localData) within an Oracle database to an external video data source.

Parameters

ctx
The source plug-in context information.

source_type
The source type of the location to where the data is to be exported.

source_location
The location to where the video data is to be exported.

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

Usage

After exporting video data, all video attributes remain unchanged and srcType, srcLocation, and srcName are updated with input values. After calling the export method, you can call the deleteContent method to delete the content of the local data.

There is no server-side native support for the export method; this method is available for user-defined sources that can support the export method.

Pragmas

none

Exception

If the value of srcType is NULL, then calling this method raises an ORDSourceExceptions.INCOMPLETE_SOURCE_INFORMATION exception.

If the export( ) method is not supported by the source plug-in being used, an ORDSourceExceptions.METHOD_NOT_SUPPORTED exception is raised.

Calling this method within a source plug-in when any other exception is raised, raises an ORDSourceExceptions.SOURCE_PLUGIN_EXCEPTION exception.

See Appendix H for more information about these exceptions.

Example

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

```
DECLARE
  obj ORDSYS.ORDVideo;
  ctx RAW(4000) :=NULL;
BEGIN
   SELECT vid INTO obj FROM TVID   WHERE N =1;
   obj.export(ctx,'FILE','VIDEODIR','x.mov');
  EXCEPTION
  WHEN ORDSYS.ORDSourceExceptions.METHOD_NOT_SUPPORTED THEN
   DBMS_OUTPUT.put_line('Source METHOD_NOT_SUPPORTED caught');
  WHEN ORDSYS.ORDSourceExceptions.SOURCE_PLUGIN_EXCEPTION THEN
   DBMS_OUTPUT.put_line('SOURCE PLUGIN EXCEPTION caught');
  WHEN ORDSYS.ORDVideoExceptions.METHOD_NOT_SUPPORTED THEN
   DBMS_OUTPUT.put_line('VIDEO METHOD_NOT_SUPPORTED EXCEPTION caught');
  WHEN ORDSYS.ORDVideoExceptions.VIDEO_PLUGIN_EXCEPTION THEN
   DBMS_OUTPUT.put_line('VIDEO PLUGIN EXCEPTION caught');
  WHEN OTHERS THEN
   DBMS_OUTPUT.put_line('OTHER EXCEPTION caught');
END;
/
```

getContentLength( ) Method

Format

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

Description

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

Parameters

ctx
The source plug-in context information.

Usage

This method is not supported for all source types. For example, "HTTP" type sources do not support this method. If you want to implement this call for "HTTP" type sources, you must define your own modified "HTTP" source type and implement this method on it.

Pragmas

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

Exception

If the value of srcType is NULL and data is not stored locally in the BLOB, then calling this method raises an exception, ORDSourceExceptions.INCOM-PLETE_SOURCE_INFORMATION.

Calling this method within a source plug-in when any other exception is raised, raises an ORDSourceExceptions.SOURCE_PLUGIN_EXCEPTION exception.

See Appendix H for more information about this exception.

Example

See the example in the import( ) Method on page 5-39.

getContentInLob( ) Method

Format

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

Description

Transfers data from a data source into the specified BLOB. The BLOB can be another BLOB, not the BLOB for the object.

Parameters

ctx
The source plug-in context information.

dest_lob
The LOB in which to receive data.

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

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

Usage

none

Pragmas

none

Exception

If the value of srcType is NULL, then calling this method raises an ORDSourceExceptions.INCOMPLETE_SOURCE_INFORMATION exception.

If the getContentInLob( ) method is not supported by the source plug-in being used, an ORDSourceExceptions.METHOD_NOT_SUPPORTED exception is raised.

Calling this method within a source plug-in when any other exception is raised, raises an ORDSourceExceptions.SOURCE_PLUGIN_EXCEPTION exception.

See Appendix H for more information about these exceptions.

Example

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(4000);
  ctx RAW(4000) :=NULL;
BEGIN
 SELECT vid INTO obj FROM TVID WHERE N = 1 ;
 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);
 -- process tempBLob
 DBMS_OUTPUT.PUT_LINE(TO_CHAR(DBMS_LOB.getLength(tempBLob)));
EXCEPTION
WHEN ORDSYS.ORDVideoExceptions.METHOD_NOT_SUPPORTED THEN
DBMS_OUTPUT.put_line('ORDVideoExceptions.METHOD_NOT_SUPPORTED caught');
WHEN OTHERS THEN
DBMS_OUTPUT.put_line('EXCEPTION caught');
END;
/
```

getContent Method

Format

```
getContent RETURN BLOB;
```

Description

Returns a handle to the local BLOB storage, that is the BLOB within the ORDVideo object.

Parameters

none

Usage

none

Pragmas

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

Exception

none

Example

A client accesses video data to be put on a Web-based player:

```
DECLARE
  obj ORDSYS.ORDVideo;
BEGIN
  SELECT vid INTO obj FROM TVID  WHERE N=1 FOR UPDATE;
  DBMS_OUTPUT.PUT_LINE('setting and getting source');
  DBMS_OUTPUT.PUT_LINE('-------------------------');
  -- import data
  obj.importFrom(ctx,'FILE','VIDEODIR','MV1.AVI');
  -- check size
  DBMS_OUTPUT.PUT_LINE(TO_CHAR(obj.getContentLength(ctx)));
  DBMS_OUTPUT.PUT_LINE(TO_CHAR(DBMS_LOB.GETLENGTH(obj.getContent)));
  DBMS_OUTPUT.PUT_LINE(obj.getSource);
  DBMS_OUTPUT.PUT_LINE('deleting contents');
```

```
  DBMS_OUTPUT.PUT_LINE('-----------------');
  obj.deleteContent;
  DBMS_OUTPUT.PUT_LINE(TO_CHAR(obj.getContentLength(ctx)));
  UPDATE TVID SET vid=obj WHERE N=1;
  COMMIT;
  EXCEPTION
        WHEN ORDSYS.ORDSourceExceptions.METHOD_NOT_SUPPORTED THEN
         DBMS_OUTPUT.put_line('Source METHOD_NOT_SUPPORTED caught');
        WHEN ORDSYS.ORDSourceExceptions.SOURCE_PLUGIN_EXCEPTION THEN
         DBMS_OUTPUT.put_line('SOURCE PLUGIN EXCEPTION caught');
        WHEN ORDSYS.ORDVideoExceptions.METHOD_NOT_SUPPORTED THEN
         DBMS_OUTPUT.put_line('VIDEO METHOD_NOT_SUPPORTED EXCEPTION caught');
        WHEN ORDSYS.ORDVideoExceptions.VIDEO_PLUGIN_EXCEPTION THEN
         DBMS_OUTPUT.put_line('VIDEO PLUGIN EXCEPTION caught');
        WHEN OTHERS THEN
         DBMS_OUTPUT.PUT_LINE('EXCEPTION Caught');
END;
/
```

deleteContent Method

Format

```
deleteContent;
```

Description

Deletes the local data from the current local source (localData).

Parameters

none

Usage

This method can be called after you export the data from the local source to an external video 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

Exception

none

Example

See the example in the import( ) Method on page 5-39.

5.2.6 ORDVideo Methods Associated with File-Like Operations

This section presents reference information on the ORDVideo methods associated with file-like operations on a data source. You can use the methods in this section specifically to manipulate video data.

openSource( ) Method

Format

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

Description

Opens a data source.

Parameters

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

ctx
The source plug-in context information. This must be allocated. You must call the openSource( ) method; see the introduction to this chapter for more information.

Usage

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

Pragmas

none

Exception

If the value for srcType is NULL and the data is not local, then calling this method raises an ORDSourceExceptions.INCOMPLETE_SOURCE_INFORMATION exception.

If the openSource( ) method is not supported by the source plug-in being used, an ORDSourceExceptions.METHOD_NOT_SUPPORTED exception is raised.

Calling this method within a source plug-in when any other exception is raised, raises an ORDSourceExceptions.SOURCE_PLUGIN_EXCEPTION exception.

See Appendix H for more information about these exceptions.

Example

Open a local data source:

```
DECLARE
  obj ORDSYS.ORDVideo;
  res INTEGER;
  ctx RAW(4000) :=NULL;
  userArg RAW(4000);
BEGIN
  select vid into obj from TVID where N =1 for UPDATE;
  res := obj.openSource(userArg, ctx);
  UPDATE TVID SET vid =obj WHERE N=1 ;
  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;
/
```

closeSource( ) Method

Format

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

Description

Closes a data source.

Parameters

ctx
The source plug-in context information. You must call the openSource( ) method; see the introduction to this chapter for more information.

Usage

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

Pragmas

none

Exception

If the value for srcType is NULL and the data is not local, then calling this method raises an ORDSourceExceptions.INCOMPLETE_SOURCE_INFORMATION exception.

If the closeSource( ) method is not supported by the source plug-in being used, an ORDSourceExceptions.METHOD_NOT_SUPPORTED exception is raised.

Calling this method within a source plug-in when any other exception is raised, raises an ORDSourceExceptions.SOURCE_PLUGIN_EXCEPTION exception.

See Appendix H for more information about these exceptions.

Example

Close an external BFILE data source:

```
DECLARE
  obj ORDSYS.ORDVideo;
  res INTEGER;
  ctx RAW(4000) :=NULL;
BEGIN
  select vid into obj from TVID where N =2 for UPDATE;
  obj.source.clearLocal;
  res := obj.closeSource(ctx);
  UPDATE TVID SET vid=obj WHERE N=2 ;
  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;
/
```

trimSource( ) Method

Format

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

Description

Trims a data source.

Parameters

ctx
The source plug-in context information. This must be allocated. You must call the openSource( ) method; see the introduction to this chapter for more information.

newlen
The trimmed new length.

Usage

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

Pragmas

none

Exception

If the value for srcType is NULL and the data is not local, then calling this method raises an ORDSourceExceptions.INCOMPLETE_SOURCE_INFORMATION exception.

If the trimSource( ) method is not supported by the source plug-in being used, an ORDSourceExceptions.METHOD_NOT_SUPPORTED exception is raised.

Calling this method within a source plug-in when any other exception is raised, raises an ORDSourceExceptions.SOURCE_PLUGIN_EXCEPTION exception.

See Appendix H for more information about these exceptions.

Example

Trim a local data source:

```
DECLARE
  obj ORDSYS.ORDVideo;
  res INTEGER;
  ctx RAW(4000) :=NULL;
BEGIN
  select vid into obj from TVID where N =1 for UPDATE;
  res := obj.trimSource(ctx,0);
  UPDATE TVID SET vid=obj WHERE N=1 ;
  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;
/
```

readFromSource( ) Method

Format

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

Description

Allows you to read a buffer of n bytes from a source beginning at a start position.

Parameters

ctx
The source plug-in context information. This must be allocated. You must call the openSource( ) method; see the introduction to this chapter for more information.

startPos
The start position in the data source.

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

buffer
The buffer into which the data will be read.

Usage

This method is not supported for HTTP sources.

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

Pragmas

none

Exception

If the value of srcType is NULL and the data is not local, then calling this method raises an ORDSourceExceptions.INCOMPLETE_SOURCE_INFORMATION exception.

If the data is local but the value of localData is NULL, then calling this method raises an ORDSourceExceptions.NULL_SOURCE exception.

If the readFromSource( ) method is not supported by the source plug-in being used, an ORDSourceExceptions.METHOD_NOT_SUPPORTED exception is raised.

Calling this method within a source plug-in when any other exception is raised, raises an ORDSourceExceptions.SOURCE_PLUGIN_EXCEPTION exception.

See Appendix H for more information about these exceptions.

Example

Read a buffer from the source:

```
DECLARE
  obj ORDSYS.ORDVideo;
  buffer RAW(4000);
  i INTEGER;
  ctx RAW(4000) :=NULL;
BEGIN
i := 20;
select vid into obj from TVID where N =1 ;
obj.readFromSource(ctx,1,i,buffer);
DBMS_OUTPUT.PUT_LINE(TO_CHAR(obj.getContentLength(ctx)));
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;
/
```

writeToSource( ) Method

Format

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

Description

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

Parameters

ctx
The source plug-in context information. This must be allocated. You must call the openSource( ) method; see the introduction to this chapter for more information.

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

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

buffer
The buffer of data to be written.

Usage

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

Pragmas

none

Exception

If the value of srcType is NULL and the data is not local, then calling this method raises an ORDSourceExceptions.INCOMPLETE_SOURCE_INFORMATION exception.

If the data is local but the localData value is NULL, then calling this method raises an ORDSourceExceptions.NULL_SOURCE exception.

If the writeToSource( ) method is not supported by the source plug-in being used, an ORDSourceExceptions.METHOD_NOT_SUPPORTED exception is raised.

Calling this method within a source plug-in when any other exception is raised, raises an ORDSourceExceptions.SOURCE_PLUGIN_EXCEPTION exception.

See Appendix H for more information about these exceptions.

Example

Write a buffer to the source:

```
DECLARE
  obj ORDSYS.ORDVideo;
  n INTEGER := 6;
  ctx RAW(4000) :=NULL;
BEGIN
  select vid into obj from TVID where N =1 for update;
  obj.writeToSource(ctx,1,n,UTL_RAW.CAST_TO_RAW('helloP'));
  DBMS_OUTPUT.PUT_LINE(TO_CHAR(obj.getContentLength(ctx)));
  update TVID set vid - obj where N = 1;
  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;
/
```

5.2.7 ORDVideo Methods Associated with the comments Attribute

This section presents reference information on the ORDVideo methods associated with the comments attribute.

appendToComments( ) Method

Format

```
appendToComments(amount IN BINARY_INTEGER,
                 buffer IN VARCHAR2);
```

Description

Appends a specified buffer and amount of comment data to the end of the comments attribute of the video object.

Parameters

amount
The amount of comment data to be appended.

buffer
The buffer of comment data to be appended.

Usage

none

Pragmas

none

Exception

The exceptions-raised behavior for this method is similar to that of the DBMS_LOB functions and procedures. See the DBMS_LOB package description in the *Oracle8i Application Developer's Reference - Packages* manual for a list of exceptions that can be raised for the DBMS_LOB functions and procedures.

Example

Append comment information to the comments attribute of the video object:

```
DECLARE
  obj ORDSYS.ORDVideo;
  i INTEGER;
  j INTEGER;
```

```
BEGIN
  SELECT vid INTO obj FROM TVID  WHERE N=1 FOR UPDATE;
  obj.writeToComments(1,18,'This is a Comments');
  obj.appendToComments(18,'This is a Comments');
  DBMS_OUTPUT.PUT_LINE(obj.readFromComments(1,obj.getCommentLength));
  DBMS_OUTPUT.PUT_LINE(obj.locateInComments('Comments',1));
  obj.trimComments(18);
  DBMS_OUTPUT.PUT_LINE(obj.readFromComments(1,18));
  i := 8;
  j := 9;
  obj.eraseFromComments(i,j);
  DBMS_OUTPUT.PUT_LINE(obj.readFromComments(1,10));
  obj.deleteComments;
  UPDATE TVID SET vid=obj WHERE N=1;
  COMMIT;
  EXCEPTION
        WHEN ORDSYS.ORDVideoExceptions.METHOD_NOT_SUPPORTED THEN
        DBMS_OUTPUT.PUT_LINE('Source not specified');
END;
/
```

writeToComments( ) Method

Format

```
writeToComments(offset IN INTEGER,
                amount IN BINARY_INTEGER,
                buffer IN VARCHAR2);
```

Description

Writes a specified amount of comment buffer data to the comments attribute of the video object beginning at the specified offset.

Parameters

offset
The starting offset position in comments where comments data is to be written.

amount
The amount of comment data to be written.

buffer
The buffer of comment data to be written.

Usage

none

Pragmas

none

Exception

The exceptions-raised behavior for this method is similar to that of the DBMS_LOB functions and procedures. See the DBMS_LOB package description in the *Oracle8i Application Developer's Reference - Packages* manual for a list of exceptions that can be raised for the DBMS_LOB functions and procedures.

Example

See the example in the appendToComments( ) Method on page 5-63.

readFromComments( ) Method

Format

```
readFromComments(offset IN INTEGER,
                 amount IN BINARY_INTEGER :=32767)
RETURN VARCHAR2;
```

Description

Reads a specified amount of comment data from the comments attribute of the video object beginning at a specified offset.

Parameters

offset
The starting offset position in comments from where comments data is to be read.

amount
The amount of comment data to be read.

Usage

none

Pragmas

Pragma RESTRICT_REFERENCES(readFromComments, WNDS, WNPS, RNDS, RNPS)

Exception

The exceptions-raised behavior for this method is similar to that of the DBMS_LOB functions and procedures. See the DBMS_LOB package description in the *Oracle8i Application Developer's Reference - Packages* manual for a list of exceptions that can be raised for the DBMS_LOB functions and procedures.

Example

See the example in the appendToComments( ) Method on page 5-63.

locateInComments( ) Method

Format

```
locateInComments(pattern    IN VARCHAR2,
                 offset     IN INTEGER := 1,
                 occurrence IN INTEGER := 1)
RETURN INTEGER;
```

Description

Matches and locates the nth occurrence of the specified pattern of character data in the comments attribute of the video object beginning at a specified offset.

Parameters

pattern
The pattern of comment data for which to search.

offset
The starting offset position in comments where the search for a match should begin.

occurrence
The nth occurrence in the comments where the pattern of comment data was found.

Usage

none

Pragmas

none

Exception

The exceptions-raised behavior for this method is similar to that of the DBMS_LOB functions and procedures. See the DBMS_LOB package description in the *Oracle8i Application Developer's Reference - Packages* manual for a list of exceptions that can be raised for the DBMS_LOB functions and procedures.

Example

See the example in the appendToComments( ) Method on page 5-63.

trimComments( ) Method

Format

```
trimComments(newlen IN INTEGER);
```

Description

Trims the length of comments of the video object to the specified new length.

Parameters

newlen
The new length to which the comments are to be trimmed.

Usage

none

Pragmas

none

Exception

The exceptions-raised behavior for this method is similar to that of the DBMS_LOB functions and procedures. See the DBMS_LOB package description in the *Oracle8i Application Developer's Reference - Packages* manual for a list of exceptions that can be raised for the DBMS_LOB functions and procedures.

Example

See the example in the appendToComments( ) Method on page 5-63.

eraseFromComments( ) Method

Format

```
eraseFromComments(amount IN OUT NOCOPY INTEGER,
                  offset IN INTEGER := 1);
```

Description

Erases a specified amount of comment data from the comments attribute of the video object beginning at a specified offset.

Parameters

amount
The amount of comment data to be erased.

offset
The starting offset position in comments where comments data is to be erased.

Usage

none

Pragmas

none

Exception

The exceptions-raised behavior for this method is similar to that of the DBMS_LOB functions and procedures. See the DBMS_LOB package description in the *Oracle8i Application Developer's Reference - Packages* manual for a list of exceptions that can be raised for the DBMS_LOB functions and procedures.

Example

See the example in the appendToComments( ) Method on page 5-63.

deleteComments( ) Method

Format

```
deleteComments;
```

Description

Deletes the comments attribute of the video object.

Parameters

none

Usage

none

Pragmas

none

Exception

The exceptions-raised behavior for this method is similar to that of the DBMS_LOB functions and procedures. See the DBMS_LOB package description in the *Oracle8i Application Developer's Reference - Packages* manual for a list of exceptions that can be raised for the DBMS_LOB functions and procedures.

Example

See the example in the appendToComments( ) Method on page 5-63.

loadCommentsFromFile( ) Method

Format

```
loadCommentsFromFile(fileobj  IN BFILE,
                     amount   IN INTEGER,
                     from_loc IN INTEGER := 1,
                     to_loc   IN INTEGER := 1);
```

Description

Loads a specified amount of comment data from a BFILE into the comments attribute of the video object beginning at a specified offset.

Parameters

fileobj
The file object to be loaded.

amount
The amount of comment data to be loaded from the BFILE.

from_loc
The location from which to load comments from the BFILE.

to_loc
The location to which to load comments.

Usage

none

Pragmas

none

Exception

The exceptions-raised behavior for this method is similar to that of the DBMS_LOB functions and procedures. See the DBMS_LOB package description in the *Oracle8i Application Developer's Reference - Packages* manual for a list of exceptions that can be raised for the DBMS_LOB functions and procedures.

Example

Load comment information from a BFILE into the comments of the video data:

```
DECLARE
  file_handle BFILE;
  obj ORDSYS.ORDVideo;
  isopen BINARY_INTEGER;
  amount INTEGER;
BEGIN
  SELECT vid INTO obj FROM TVID  WHERE N=1 FOR UPDATE;
  --file_handle := BFILENAME(obj.getSourceLocation, obj.getSourceName);
  file_handle := BFILENAME('VIDEODIR', 'testvid.dat');
     isopen := DBMS_LOB.FILEISOPEN(file_handle);
     IF isopen = 0 THEN
       --dbms_output.put_line('File Not Open');
       DBMS_LOB.FILEOPEN(file_handle, DBMS_LOB.FILE_READONLY);
     END IF;
     --dbms_output.put_line('File is now Open');
     isopen := DBMS_LOB.FILEISOPEN(file_handle);
     IF isopen <> 0 THEN
       amount := DBMS_LOB.GETLENGTH(file_handle);
     END IF;
     obj.loadCommentsFromFile(file_handle, 18, 1, 18);
     dbms_output.put_line(obj.getCommentLength);
  UPDATE TVID SET vid=obj WHERE N=1;
  COMMIT;
  EXCEPTION
        WHEN ORDSYS.ORDVideoExceptions.METHOD_NOT_SUPPORTED THEN
         DBMS_OUTPUT.PUT_LINE('Source not specified');
END;
/
```

copyCommentsOut( ) Method

Format

```
copyCommentsOut(dest     IN OUT NOCOPY CLOB,
                amount   IN INTEGER,
                from_loc IN INTEGER := 1,
                to_loc   IN INTEGER := 1);
```

Description

Copies a specified amount of video object comments attribute into the given CLOB.

Parameters

dest
The destination to which the comments are to be copied.

amount
The amount of comments data to be copied.

from_loc
The location from which to copy the comments.

to_loc
The location to which to copy the comments.

Usage

none

Pragmas

none

Exception

The exceptions-raised behavior for this method is similar to that of the DBMS_LOB functions and procedures. See the DBMS_LOB package description in the *Oracle8i Application Developer's Reference - Packages* manual for a list of exceptions that can be raised for the DBMS_LOB functions and procedures.

Example

Copy comments of the video data to the given CLOB:

```
DECLARE
  file_handle BFILE;
  obj ORDSYS.ORDVideo;
  obj1 ORDSYS.ORDVideo;
BEGIN
  SELECT vid INTO obj1 FROM TVID  WHERE N=2 FOR UPDATE;
  SELECT vid INTO obj FROM TVID  WHERE N=1;
  obj.copyCommentsOut(obj1.comments,obj.getCommentLength,1,10);
  DBMS_OUTPUT.put_line(obj1.getCommentLength);
  DBMS_OUTPUT.put_line(obj.getCommentLength);
  UPDATE TVID SET vid=obj1 WHERE N=2;
  COMMIT;
  EXCEPTION
        WHEN ORDSYS.ORDVideoExceptions.METHOD_NOT_SUPPORTED THEN
        DBMS_OUTPUT.PUT_LINE('Source not specified');
END;
/
```

compareComments( ) Method

Format

```
compareComments(compare_with_lob        IN CLOB,
                amount                  IN INTEGER := 4294967295,
                starting_pos_in_comment IN INTEGER := 1,
                starting_pos_in_compare IN INTEGER := 1)
RETURN INTEGER;
```

Description

Compares a specified amount of comments of video data with comments of the other CLOB provided.

Parameters

compare_with_lob
The comparison comments.

amount
The amount of comments of video data to compare with the comparison comments.

starting_pos_in_comment
The starting position in the comments attribute of the video object.

starting_pos_in_compare
The starting position in the comparison comments.

Usage

none

Pragmas

Pragma RESTRICT_REFERENCES(compareComments, WNDS, WNPS, RNDS, RNPS)

Exception

The exceptions-raised behavior for this method is similar to that of the DBMS_LOB functions and procedures. See the DBMS_LOB package description in the *Oracle8i*

Application Developer's Reference - Packages manual for a list of exceptions that can be raised for the DBMS_LOB functions and procedures.

Example

Compare comments of the video data with comments of another CLOB:

```
DECLARE
  file_handle BFILE;
  obj ORDSYS.ORDVideo;
  obj1 ORDSYS.ORDVideo;
BEGIN
  SELECT vid INTO obj FROM TVID  WHERE N=2 ;
  SELECT vid INTO obj1 FROM TVID  WHERE N=1;
  DBMS_OUTPUT.put_line('comparison output');
  DBMS_OUTPUT.put_line(obj.compareComments(obj1.comments,obj.getCommentLength,
  1,18));
  EXCEPTION
        WHEN ORDSYS.ORDVideoExceptions.METHOD_NOT_SUPPORTED THEN
        DBMS_OUTPUT.PUT_LINE('Source not specified');
END;
/
```

getCommentLength( ) Method

Format

```
getCommentLength RETURN INTEGER;
```

Description

Returns the length of the comments attribute of the video object.

Parameters

none

Usage

none

Pragmas

Pragma RESTRICT_REFERENCES(getCommentLength, WNDS, WNPS, RNDS, RNPS)

Exception

The exceptions-raised behavior for this method is similar to that of the DBMS_LOB functions and procedures. See the DBMS_LOB package description in the *Oracle8i Application Developer's Reference - Packages* manual for a list of exceptions that can be raised for the DBMS_LOB functions and procedures.

Example

See the example in the compareComments( ) Method on page 5-76.

5.2.8 ORDVideo Methods Associated with Video Attributes Accessors

This section presents reference information on the ORDVideo methods associated with the video attributes accessors.

setFormat( ) Method

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 audio object.

Usage

Calling this method implicitly calls the setUpdateTime( ) method.

Pragmas

none

Exception

If the value for the knownFormat parameter is NULL, then calling this method raises a NULL_INPUT_VALUE exception.

Example

Set the format for some stored video data:

```
DECLARE
  obj ORDSYS.ORDVideo;
BEGIN
  SELECT vid INTO obj FROM TVID  WHERE N=1 FOR UPDATE;
  DBMS_OUTPUT.PUT_LINE('writing format');
  DBMS_OUTPUT.PUT_LINE('--------------');
  obj.setFormat('avi');
  DBMS_OUTPUT.PUT_LINE(obj.getFormat);
  UPDATE TVID SET vid=obj WHERE N=1;
  COMMIT;
END;
/
```

getFormat Method

Format

```
getFormat RETURN VARCHAR2;
```

Description

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

Parameters

none

Usage

none

Pragmas

Pragma RESTRICT_REFERENCES(getStoredFormat, WNDS, WNPS, RNDS, RNPS)

Exception

If the value for format is NULL, then calling this method raises a VIDEO_FORMAT_IS_NULL exception.

Example

See the example in the setFormat( ) Method on page 5-79.

getFormat( ) Method

Format

```
getFormat(ctx IN OUT RAW) RETURN VARCHAR2;
```

Description

Calls the format plug-in to read the format embedded in the stored video data.

Parameters

ctx
The format plug-in context information.

Usage

If the format found in the object is NULL, then the getFormat( ) method uses the default format plug-in to read the video data to determine the format; otherwise, it uses your user-defined format plug-in.

Video file format information can be extracted from the formatted video data itself. You can extend support to a file format not known by the ORDVideo object by implementing an ORDPLUGINS.ORDX_<format>_VIDEO package that supports that file format. See Section 2.3.13 for more information.

Pragmas

none

Exception

If the video plug-in raises an exception when calling this method, then the getFormat( ) method raises a VIDEO_PLUGIN_EXCEPTION exception.

Example

Return the file format for video data stored in the database:

```
DECLARE
  obj ORDSYS.ORDVideo;
  res VARCHAR2(4000);
  ctx RAW(4000) :=NULL;
```

```
BEGIN
  SELECT vid INTO obj FROM TVID  WHERE N=1 FOR UPDATE;
  res := obj.getFormat(ctx);
  DBMS_OUTPUT.put_line(res );
  EXCEPTION
        WHEN ORDSYS.ORDSourceExceptions.METHOD_NOT_SUPPORTED THEN
         DBMS_OUTPUT.put_line('Source METHOD_NOT_SUPPORTED caught');
        WHEN ORDSYS.ORDSourceExceptions.SOURCE_PLUGIN_EXCEPTION THEN
         DBMS_OUTPUT.put_line('SOURCE PLUGIN EXCEPTION caught');
        WHEN ORDSYS.ORDVideoExceptions.METHOD_NOT_SUPPORTED THEN
         DBMS_OUTPUT.put_line('VIDEO METHOD_NOT_SUPPORTED EXCEPTION caught');
        WHEN ORDSYS.ORDVideoExceptions.VIDEO_PLUGIN_EXCEPTION THEN
         DBMS_OUTPUT.put_line('VIDEO PLUGIN EXCEPTION caught');
        WHEN OTHERS THEN
         DBMS_OUTPUT.PUT_LINE('method not supported');
END;
/
```

setFrameSize( ) Method

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

Calling this method implicitly calls the setUpdateTime( ) method.

Pragmas

none

Exception

If the value for either the knownWidth or knownHeight parameters is NULL, then calling this method raises a NULL_INPUT_VALUE exception.

Example

Set the frame size for video data:

```
DECLARE
  obj ORDSYS.ORDVideo;
BEGIN
  select vid into obj from TVID where N =1 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 TVID set vid = obj where N = 1;
  COMMIT;
END;
/
```

getFrameSize( ) Method

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

none

Pragmas

Pragma RESTRICT_REFERENCES(getFrameSize, WNDS, WNPS, RNDS, RNPS)

Exception

none

Example

Return the frame size for video data:

```
DECLARE
  obj ORDSYS.ORDVideo;
  width INTEGER;
  height INTEGER;
BEGIN
  SELECT vid INTO obj FROM TVID  WHERE N=1 ;
  obj.getFrameSize(width, height);
  DBMS_OUTPUT.put_line('width :' || width);
```

```
  DBMS_OUTPUT.put_line('height :' || height);
END;
/
```

getFrameSize( ) Method

Format

```
getFrameSize(
             ctx    IN OUT RAW,
             width  OUT INTEGER,
             height OUT INTEGER);
```

Description

Calls the format plug-in to read the frame size embedded in the stored video data.

Parameters

ctx
The format plug-in context information.

width
The frame width in pixels.

height
The frame height in pixels.

Usage

The video frame size information is available from the header of the formatted video data.

If the format found in the object is NULL, then the getFrameSize( ) method uses the default format plug-in to read the video data to determine the frame size; otherwise, it uses your user-defined format plug-in.

Video frame size information can be extracted from the video data itself. You can extend support to a format that is not understood by the ORDVideo object by implementing an ORDPLUGINS.ORDX_<format>_VIDEO package that supports that format. See Section 2.3.13 for more information.

Pragmas

none

Exception

If the video plug-in raises an exception when calling this method, then it raises a VIDEO_PLUGIN_EXCEPTION exception.

Example

Call the format plug-in to read the actual frame size embedded in the stored video data:

```
DECLARE
  obj ORDSYS.ORDVideo;
  width VARCHAR2(4000);
  height VARCHAR2(4000);
  ctx RAW(4000) :=NULL;
BEGIN
  SELECT vid INTO obj FROM TVID  WHERE N=1 FOR UPDATE;
  obj.getFrameSize(ctx,width, height);
  EXCEPTION
        WHEN ORDSYS.ORDSourceExceptions.METHOD_NOT_SUPPORTED THEN
         DBMS_OUTPUT.put_line('Source METHOD_NOT_SUPPORTED caught');
        WHEN ORDSYS.ORDSourceExceptions.SOURCE_PLUGIN_EXCEPTION THEN
         DBMS_OUTPUT.put_line('SOURCE PLUGIN EXCEPTION caught');
        WHEN ORDSYS.ORDVideoExceptions.METHOD_NOT_SUPPORTED THEN
         DBMS_OUTPUT.put_line('VIDEO METHOD_NOT_SUPPORTED EXCEPTION caught');
        WHEN ORDSYS.ORDVideoExceptions.VIDEO_PLUGIN_EXCEPTION THEN
         DBMS_OUTPUT.put_line('VIDEO PLUGIN EXCEPTION caught');
        WHEN OTHERS THEN
         DBMS_OUTPUT.PUT_LINE('method not supported');
END;
/
```

setFrameResolution( ) Method

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

Calling this method implicitly calls the setUpdateTime( ) method.

Pragmas

none

Exception

If the value for the knownFrameResolution parameter is NULL, then calling this method raises a NULL_INPUT_VALUE exception.

Example

See the example in the setFrameSize( ) Method on page 5-83.

getFrameResolution Method

Format

```
getFrameResolution RETURN INTEGER;
```

Description

Returns the value of the frameResolution attribute of the video object.

Parameters

none

Usage

none

Pragmas

Pragma RESTRICT_REFERENCES(getFrameResolution, WNDS, WNPS, RNDS, RNPS)

Exception

none

Example

Return the value of the frame resolution for the video data:

```
DECLARE
  obj ORDSYS.ORDVideo;
  res INTEGER;
BEGIN
  SELECT vid INTO obj FROM TVID  WHERE N=1 ;
  res := obj.getFrameResolution;
  DBMS_OUTPUT.put_line('resolution : ' ||res);
END;
/
```

getFrameResolution( ) Method

Format

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

Description

Calls the format plug-in to read the frame resolution embedded in the stored video data.

Parameters

ctx
The format plug-in context information.

Usage

The video frame resolution information is available from the header of the formatted video data.

If the format found in the object is NULL, then the getFrameResolution( ) method uses the default format plug-in to read the video data to determine the frame resolution; otherwise, it uses your user-defined format plug-in.

Video frame resolution information can be extracted from the video data itself. You can extend support to a format not understood by the ORDVideo object by implementing an ORDPLUGINS.ORDX_<format>_VIDEO package that supports that format. See Section 2.3.13 for more information.

Pragmas

none

Exception

If the video plug-in raises an exception when calling this method, then it raises a VIDEO_PLUGIN_EXCEPTION exception.

Example

Call the format plug-in to read the actual frame resolution embedded in the stored video data:

```
DECLARE
  obj ORDSYS.ORDVideo;
  res INTEGER;
  ctx RAW(4000) :=NULL;
BEGIN
  SELECT vid INTO obj FROM TVID  WHERE N=1 FOR UPDATE;
  res := obj.getFrameResolution(ctx);
  DBMS_OUTPUT.put_line(res );
  EXCEPTION
        WHEN ORDSYS.ORDSourceExceptions.METHOD_NOT_SUPPORTED THEN
         DBMS_OUTPUT.put_line('Source METHOD_NOT_SUPPORTED caught');
        WHEN ORDSYS.ORDSourceExceptions.SOURCE_PLUGIN_EXCEPTION THEN
         DBMS_OUTPUT.put_line('SOURCE PLUGIN EXCEPTION caught');
        WHEN ORDSYS.ORDVideoExceptions.METHOD_NOT_SUPPORTED THEN
         DBMS_OUTPUT.put_line('VIDEO METHOD_NOT_SUPPORTED EXCEPTION caught');
        WHEN ORDSYS.ORDVideoExceptions.VIDEO_PLUGIN_EXCEPTION THEN
         DBMS_OUTPUT.put_line('VIDEO PLUGIN EXCEPTION caught');
        WHEN OTHERS THEN
         DBMS_OUTPUT.PUT_LINE('method not supported');
END;
/
```

setFrameRate( ) Method

Format

```
setFrameRate(knownFrameRate IN INTEGER);
```

Description

Sets the value of the frameRate attribute of the video object.

Parameters

knownFrameRate
The frame rate.

Usage

Calling this method implicitly calls the setUpdateTime( ) method.

Pragmas

none

Exception

If the value for the knownFrameRate parameter is NULL, then calling this method raises a NULL_INPUT_VALUE exception.

Example

See the example in the setFrameSize( ) Method on page 5-83.

getFrameRate Method

Format

```
getFrameRate RETURN INTEGER;
```

Description

Returns the value of the frameRate attribute of the video object.

Parameters

none

Usage

none

Pragmas

Pragma RESTRICT_REFERENCES(getFrameRate, WNDS, WNPS, RNDS, RNPS)

Exception

none

Example

Return the object attribute value of the frame rate for video data stored in the database:

```
DECLARE
  obj ORDSYS.ORDVideo;
  res INTEGER;
BEGIN
  SELECT vid INTO obj FROM TVID  WHERE N=1 ;
  res := obj.getFrameRate;
  DBMS_OUTPUT.put_line('frame rate : ' ||res);
END;
/
```

getFrameRate( ) Method

Format

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

Description

Calls the format plug-in to read the frame rate embedded in the stored video data.

Parameters

ctx
The format plug-in context information.

Usage

The video frame rate information is available from the header of the formatted video data.

If the format found in the object is NULL, then the getFrameRate( ) method uses the default format plug-in to read the video data to determine the frame rate; otherwise, it uses your user-defined format plug-in.

Video frame rate information can be extracted from the video data itself. You can extend support to a format not understood by the ORDVideo object by implementing an ORDPLUGINS.ORDX_<format>_VIDEO package that supports that format. See Section 2.3.13 for more information.

Pragmas

none

Exception

If the video plug-in raises an exception when calling this method, then it raises a VIDEO_PLUGIN_EXCEPTION exception.

Example

Call the format plug-in to read the actual frame rate embedded in the stored audio data:

```
DECLARE
```

```
  obj ORDSYS.ORDVideo;
  res INTEGER;
  ctx RAW(4000) :=NULL;
BEGIN
  SELECT vid INTO obj FROM TVID  WHERE N=1 FOR UPDATE;
  res := obj.getFrameRate(ctx);
  DBMS_OUTPUT.put_line(res );
  EXCEPTION
        WHEN ORDSYS.ORDSourceExceptions.METHOD_NOT_SUPPORTED THEN
         DBMS_OUTPUT.put_line('Source METHOD_NOT_SUPPORTED caught');
        WHEN ORDSYS.ORDSourceExceptions.SOURCE_PLUGIN_EXCEPTION THEN
         DBMS_OUTPUT.put_line('SOURCE PLUGIN EXCEPTION caught');
        WHEN ORDSYS.ORDVideoExceptions.METHOD_NOT_SUPPORTED THEN
         DBMS_OUTPUT.put_line('VIDEO METHOD_NOT_SUPPORTED EXCEPTION caught');
        WHEN ORDSYS.ORDVideoExceptions.VIDEO_PLUGIN_EXCEPTION THEN
         DBMS_OUTPUT.put_line('VIDEO PLUGIN EXCEPTION caught');
        WHEN OTHERS THEN
         DBMS_OUTPUT.PUT_LINE('method not supported');
END;
/
```

setVideoDuration( ) Method

Format

```
setVideoDuration(knownVideoDuration RETURN INTEGER);
```

Description

Sets the value of the videoDuration attribute of the video object.

Parameters

knownVideoDuration
A known video duration.

Usage

Calling this method implicitly calls the setUpdateTime( ) method.

Pragmas

none

Exception

If the value for the knownVideoDuration parameter is NULL, then calling this method raises a NULL_INPUT_VALUE exception.

Example

See the example in the setFrameSize( ) Method on page 5-83.

getVideoDuration Method

Format

```
getVideoDuration RETURN INTEGER;
```

Description

Returns the value of the videoDuration attribute of the video object.

Parameters

none

Usage

none

Pragmas

Pragma RESTRICT_REFERENCES(getVideoDuration, WNDS, WNPS, RNDS, RNPS)

Exception

none

Example

Return the total time to play the video data:

```
DECLARE
  obj ORDSYS.ORDVideo;
  res INTEGER;
BEGIN
  SELECT vid INTO obj FROM TVID  WHERE N=1 ;
  res := obj.getVideoDuration;
  DBMS_OUTPUT.put_line('video duration : ' ||res);
END;
/
```

getVideoDuration( ) Method

Format

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

Description

Calls the format plug-in to read the video duration embedded in the stored video data.

Parameters

ctx
The format plug-in context information.

Usage

The video duration information is available from the header of the formatted video data.

If the format found in the object is NULL, then the getVideoDuration( ) method uses the default format plug-in to read the video data to determine the video duration; otherwise, it uses your user-defined format plug-in.

Video duration information can be extracted from the video data itself. You can extend support to a format that is not understood by the ORDVideo object by implementing an ORDPLUGINS.ORDX_<format>_VIDEO package that supports that format. See Section 2.3.13 for more information.

Pragmas

none

Exception

If the video plug-in raises an exception when calling this method, then it raises a VIDEO_PLUGIN_EXCEPTION exception.

Example

Calls the format plug-in to read the actual video duration embedded in the stored video data:

```
DECLARE
  obj ORDSYS.ORDVideo;
  res INTEGER;
  ctx RAW(4000) :=NULL;
BEGIN
  SELECT vid INTO obj FROM TVID  WHERE N=1 FOR UPDATE;
  res := obj.getVideoDuration(ctx);
  --DBMS_OUTPUT.put_line(res );
  EXCEPTION
        WHEN ORDSYS.ORDSourceExceptions.METHOD_NOT_SUPPORTED THEN
         DBMS_OUTPUT.put_line('Source METHOD_NOT_SUPPORTED caught');
        WHEN ORDSYS.ORDSourceExceptions.SOURCE_PLUGIN_EXCEPTION THEN
         DBMS_OUTPUT.put_line('SOURCE PLUGIN EXCEPTION caught');
        WHEN ORDSYS.ORDVideoExceptions.METHOD_NOT_SUPPORTED THEN
         DBMS_OUTPUT.put_line('VIDEO METHOD_NOT_SUPPORTED EXCEPTION caught');
        WHEN ORDSYS.ORDVideoExceptions.VIDEO_PLUGIN_EXCEPTION THEN
         DBMS_OUTPUT.put_line('VIDEO PLUGIN EXCEPTION caught');
        WHEN OTHERS THEN
         DBMS_OUTPUT.PUT_LINE('method not supported');
END;
/
```

setNumberOfFrames( ) Method

Format

```
setNumberOfFrames(knownNumberOfFrames RETURN INTEGER);
```

Description

Sets the value of the numberOfFrames attribute of the video object.

Parameters

knownNumberOfFrames
A known number of frames.

Usage

Calling this method implicitly calls the setUpdateTime( ) method.

Pragmas

none

Exception

If the value for the knownNumberOfFrames parameter is NULL, then calling this method raises a NULL_INPUT_VALUE exception.

Example

See the example in the setFrameSize( ) Method on page 5-83.

getNumberOfFrames Method

Format

```
getNumberOfFrames RETURN INTEGER;
```

Description

Returns the value of the numberOfFrames attribute of the video object.

Parameters

none

Usage

none

Pragmas

Pragma RESTRICT_REFERENCES(getNumberOfFrames, WNDS, WNPS, RNDS, RNPS)

Exception

none

Example

Return the object attribute value of the total number of frames in the video data:

```
DECLARE
  obj ORDSYS.ORDVideo;
  res INTEGER;
BEGIN
  SELECT vid INTO obj FROM TVID  WHERE N=1 ;
  res := obj.getNumberOfFrames;
  DBMS_OUTPUT.put_line('number of frames : ' ||res);
END;
/
```

getNumberOfFrames( ) Method

Format

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

Description

Calls the format plug-in to read the number of frames embedded in the stored video data.

Parameters

ctx
The format plug-in context information.

Usage

The total number of frames information is available from the header of the formatted video data.

If the format found in the object is NULL, then the getNumberOfFrames( ) method uses the default format plug-in to read the video data to determine the number of frames; otherwise, it uses your user-defined format plug-in.

Total number of frames information can be extracted from the video data itself. 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 Section 2.3.13 for more information.

Pragmas

none

Exception

If the video plug-in raises an exception when calling this method, then it raises a VIDEO_PLUGIN_EXCEPTION exception.

Example

Call the format plug-in to read the actual number of frames embedded in the stored video data:

```
DECLARE
  obj ORDSYS.ORDVideo;
  res INTEGER;
  ctx RAW(4000) :=NULL;
BEGIN
  SELECT vid INTO obj FROM TVID  WHERE N=1 FOR UPDATE;
  res := obj.getNumberOfFrames(ctx);
  DBMS_OUTPUT.put_line(res );
  EXCEPTION
        WHEN ORDSYS.ORDSourceExceptions.METHOD_NOT_SUPPORTED THEN
         DBMS_OUTPUT.put_line('Source METHOD_NOT_SUPPORTED caught');
        WHEN ORDSYS.ORDSourceExceptions.SOURCE_PLUGIN_EXCEPTION THEN
         DBMS_OUTPUT.put_line('SOURCE PLUGIN EXCEPTION caught');
        WHEN ORDSYS.ORDVideoExceptions.METHOD_NOT_SUPPORTED THEN
         DBMS_OUTPUT.put_line('VIDEO METHOD_NOT_SUPPORTED EXCEPTION caught');
        WHEN ORDSYS.ORDVideoExceptions.VIDEO_PLUGIN_EXCEPTION THEN
         DBMS_OUTPUT.put_line('VIDEO PLUGIN EXCEPTION caught');
        WHEN OTHERS THEN
         DBMS_OUTPUT.PUT_LINE('method not supported');
END;
/
```

setCompressionType( ) Method

Format

```
setCompressionType(knownCompressionType IN VARCHAR2);
```

Description

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

Parameters

knownCompressionType
A known compression type.

Usage

Calling this method implicitly calls the setUpdateTime( ) method.

Pragmas

none

Exception

If the value for the knownCompressionType parameter is NULL, then calling this method raises a NULL_INPUT_VALUE exception.

Example

See the example in the setFrameSize( ) Method on page 5-83.

getCompressionType Method

Format

```
getCompressionType RETURN VARCHAR2;
```

Description

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

Parameters

none

Usage

none

Pragmas

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

Exception

none

Example

Return the object attribute value of the compressionType attribute of the video object:

```
DECLARE
  obj ORDSYS.ORDVideo;
  res VARCHAR2(4000);
BEGIN
  SELECT vid INTO obj FROM TVID  WHERE N=1 ;
  res := obj.getCompressionType;
  DBMS_OUTPUT.put_line('compression type: ' ||res);
END;
/
```

getCompressionType( ) Method

Format

```
getCompressionType(ctx IN OUT RAW) RETURN VARCHAR2;
```

Description

Calls the format plug-in to read the compression type embedded in the stored video data.

Parameters

ctx
The format plug-in context information.

Usage

The video compression type information is available from the header of the formatted video data.

If the format found in the object is NULL, then the getCompressionType( ) method uses the default format plug-in to read the video data to determine the compression type; otherwise, it uses your user-defined format plug-in.

Video compression type information can be extracted from the audio data itself. 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 Section 2.3.13 for more information.

Pragmas

none

Exception

If the audio plug-in raises an exception when calling this method, then it raises a VIDEO_PLUGIN_EXCEPTION exception.

Example

Call the format plug-in to read the actual compression type embedded in the stored video data:

```
DECLARE
  obj ORDSYS.ORDVideo;
  res VARCHAR2(4000);
  ctx RAW(4000) :=NULL;
BEGIN
  SELECT vid INTO obj FROM TVID  WHERE N=1 FOR UPDATE;
  res := obj.getCompressionType(ctx);
  DBMS_OUTPUT.put_line(res );
  EXCEPTION
        WHEN ORDSYS.ORDSourceExceptions.METHOD_NOT_SUPPORTED THEN
        DBMS_OUTPUT.put_line('Source METHOD_NOT_SUPPORTED caught');
        WHEN ORDSYS.ORDSourceExceptions.SOURCE_PLUGIN_EXCEPTION THEN
        DBMS_OUTPUT.put_line('SOURCE PLUGIN EXCEPTION caught');
        WHEN ORDSYS.ORDVideoExceptions.METHOD_NOT_SUPPORTED THEN
        DBMS_OUTPUT.put_line('VIDEO METHOD_NOT_SUPPORTED EXCEPTION caught');
        WHEN ORDSYS.ORDVideoExceptions.VIDEO_PLUGIN_EXCEPTION THEN
        DBMS_OUTPUT.put_line('VIDEO PLUGIN EXCEPTION caught');
        WHEN OTHERS THEN
        DBMS_OUTPUT.PUT_LINE('method not supported');
END;
/
```

setNumberOfColors( ) Method

Format

```
setNumberOfColors(knownNumberOfColors RETURN INTEGER);
```

Description

Sets the value of the numberOfColors attribute of the video object.

Parameters

knownNumberOfColors
A known number of colors.

Usage

Calling this method implicitly calls the setUpdateTime( ) method.

Pragmas

none

Exception

If the value for the knownNumberOfColors parameter is NULL, then calling this method raises a NULL_INPUT_VALUE exception.

Example

See the example in the setFrameSize( ) Method on page 5-83.

getNumberOfColors Method

Format

```
getNumberOfColors RETURN INTEGER;
```

Description

Returns the value of the numberOfColors attribute of the video object.

Parameters

none

Usage

none

Pragmas

Pragma RESTRICT_REFERENCES(getNumberOfColors, WNDS, WNPS, RNDS, RNPS)

Exception

none

Example

Return the object attribute value of the numberOfColors attribute of the video object:

```
DECLARE
  obj ORDSYS.ORDVideo;
  res INTEGER;
BEGIN
  SELECT vid INTO obj FROM TVID  WHERE N=1 ;
  res := obj.getNumberOfColors;
  DBMS_OUTPUT.put_line('number of colors: ' ||res);
END;
/
```

getNumberOfColors( ) Method

Format

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

Description

Calls the format plug-in to read the number of colors embedded in the stored video data.

Parameters

ctx
The format plug-in context information.

Usage

The total number of colors information is available from the header of the formatted video data.

If the format found in the object is NULL, then the getNumberOfColors( ) method uses the default format plug-in to read the video data to determine the number of colors; otherwise, it uses your user-defined format plug-in.

Total number of colors information can be extracted from the video data itself. You can extend support to a format that is not understood by the ORDVideo object by implementing an ORDPLUGINS.ORDX_<format>_VIDEO package that supports that format. See Section 2.3.13 for more information.

Pragmas

none

Exception

If the video plug-in raises an exception when calling this method, then it raises a VIDEO_PLUGIN_EXCEPTION exception.

Example

Calls the format plug-in to read the actual number of colors embedded in the stored video data:

```
DECLARE
  obj ORDSYS.ORDVideo;
  res INTEGER;
  ctx RAW(4000) :=NULL;
BEGIN
  SELECT vid INTO obj FROM TVID  WHERE N=1 FOR UPDATE;
  res := obj.getNumberOfColors(ctx);
  DBMS_OUTPUT.put_line(res );
  EXCEPTION
        WHEN ORDSYS.ORDSourceExceptions.METHOD_NOT_SUPPORTED THEN
        DBMS_OUTPUT.put_line('Source METHOD_NOT_SUPPORTED caught');
        WHEN ORDSYS.ORDSourceExceptions.SOURCE_PLUGIN_EXCEPTION THEN
        DBMS_OUTPUT.put_line('SOURCE PLUGIN EXCEPTION caught');
        WHEN ORDSYS.ORDVideoExceptions.METHOD_NOT_SUPPORTED THEN
        DBMS_OUTPUT.put_line('VIDEO METHOD_NOT_SUPPORTED EXCEPTION caught');
        WHEN ORDSYS.ORDVideoExceptions.VIDEO_PLUGIN_EXCEPTION THEN
        DBMS_OUTPUT.put_line('VIDEO PLUGIN EXCEPTION caught');
        WHEN OTHERS THEN
        DBMS_OUTPUT.PUT_LINE('method not supported');
END;
/
```

setBitRate( ) Method

Format

```
setBitRate(knownBitRate IN INTEGER);
```

Description

Sets the value of the bitRate attribute of the video object.

Parameters

knownBitRate
The bit rate.

Usage

Calling this method implicitly calls the setUpdateTime( ) method.

Pragmas

none

Exception

If the value for the knownBitRate parameter is NULL, then calling this method raises a NULL_INPUT_VALUE exception.

Example

See the example in the setFrameSize( ) Method on page 5-83.

getBitRate Method

Format

```
getBitRate RETURN INTEGER;
```

Description

Returns the value of the bitRate attribute of the video object.

Parameters

none

Usage

none

Pragmas

Pragma RESTRICT_REFERENCES(getBitRate, WNDS, WNPS, RNDS, RNPS)

Exception

none

Example

Return the object attribute value of the bitRate attribute of the video object:

```
DECLARE
  obj ORDSYS.ORDVideo;
  res INTEGER;
BEGIN
  SELECT vid INTO obj FROM TVID  WHERE N=1 ;
  res := obj.getBitRate;
  DBMS_OUTPUT.put_line('bit rate : ' || res );
END;
/
```

getBitRate( ) Method

Format

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

Description

Calls the format plug-in to read the bit rate embedded in the stored video data.

Parameters

ctx
The format plug-in context information.

Usage

The video bit rate information is available from the header of the formatted video data.

If the format found in the object is NULL, then the getBitRate( ) method uses the default format plug-in to read the video data to determine the bit rate; otherwise, it uses your user-defined format plug-in.

Video bit rate information can be extracted from the video data itself. 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 Section 2.3.13 for more information.

Pragmas

none

Exception

If the video plug-in raises an exception when calling this method, then it raises a VIDEO_PLUGIN_EXCEPTION exception.

Example

Call the format plug-in to read the actual bit rate embedded in the stored video data:

```
DECLARE
```

```
  obj ORDSYS.ORDVideo;
  res INTEGER;
  ctx RAW(4000) :=NULL;
BEGIN
  SELECT vid INTO obj FROM TVID  WHERE N=1 FOR UPDATE;
  res := obj.getBitRate(ctx);
  DBMS_OUTPUT.put_line(res );
  EXCEPTION
        WHEN ORDSYS.ORDSourceExceptions.METHOD_NOT_SUPPORTED THEN
        DBMS_OUTPUT.put_line('Source METHOD_NOT_SUPPORTED caught');
        WHEN ORDSYS.ORDSourceExceptions.SOURCE_PLUGIN_EXCEPTION THEN
        DBMS_OUTPUT.put_line('SOURCE PLUGIN EXCEPTION caught');
        WHEN ORDSYS.ORDVideoExceptions.METHOD_NOT_SUPPORTED THEN
        DBMS_OUTPUT.put_line('VIDEO METHOD_NOT_SUPPORTED EXCEPTION caught');
        WHEN ORDSYS.ORDVideoExceptions.VIDEO_PLUGIN_EXCEPTION THEN
        DBMS_OUTPUT.put_line('VIDEO PLUGIN EXCEPTION caught');
        WHEN OTHERS THEN
        DBMS_OUTPUT.PUT_LINE('method not supported');
END;
/
```

setKnownAttributes( ) Method

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

Calling this method implicitly calls the setUpdateTime( ) method.

Pragmas

none

Exception

none

Example

Set the property information for all known attributes for video data:

```
DECLARE
  obj ORDSYS.ORDVideo;
BEGIN
  select vid into obj from TVID where N =1 for update;
  obj.setKnownAttributes('MOOV',1,2,4,5,20,8,'Cinepak', 256, 1500);
  DBMS_OUTPUT.put_line('width: '  || TO_CHAR(obj.width));
  DBMS_OUTPUT.put_line('height: ' || TO_CHAR(obj.height));
  DBMS_OUTPUT.put_line('format: ' || obj.getFormat);
  DBMS_OUTPUT.put_line('frame resolution: ' ||TO_CHAR(obj.getFrameResolution));
  DBMS_OUTPUT.put_line('frame rate: ' || TO_CHAR(obj.getFrameRate));
  DBMS_OUTPUT.put_line('video duration: ' || TO_CHAR(obj.getVideoDuration));
  DBMS_OUTPUT.put_line('number of frames: ' || TO_CHAR(obj.getNumberOfFrames));
  DBMS_OUTPUT.put_line('compression type: ' || obj.getCompressionType);
  DBMS_OUTPUT.put_line('bite rate: ' || TO_CHAR(obj.getBitRate));
  DBMS_OUTPUT.put_line('number of colors: ' || TO_CHAR(obj.getNumberOfColors));
  update TVID set vid = obj where N = 1;
  COMMIT;
END;
/
```

setProperties( ) Method

Format

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

Description

Reads the video data to get the values of the object attributes and then stores them in the object. For the known attributes that ORDVideo understands, it sets the properties for these attributes, which include: format, frame size, frame resolution, frame rate, video duration, number of frames, compression type, number of colors, and bit rate.

Parameters

ctx
The format plug-in context information.

Usage

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

Exception

If the video plug-in raises an exception when calling this method, then it raises a VIDEO_PLUGIN_EXCEPTION exception.

Example

Set the property information for known video attributes:

```
DECLARE
  obj ORDSYS.ORDVideo;
  ctx RAW(4000) :=NULL;
BEGIN
  select vid into obj from TVID where N =1 for update;
  obj.setProperties(ctx);
```

```
  update TVID set vid = obj where N = 1;
  COMMIT;
  EXCEPTION
    WHEN OTHERS THEN
     DBMS_OUTPUT.put_line('exception raised');
END;
/
```

checkProperties( ) Method

Format

```
checkProperties(ctx IN OUT RAW) RETURN BOOLEAN;
```

Description

Checks all the properties of the stored video data, including the following video attributes: format, frame size, frame resolution, frame rate, video duration, number of frames, compression type, number of colors, and bit rate.

Parameters

ctx
The format plug-in context information.

Usage

If the format is set to NULL, then the checkProperties( ) method uses the default format plug-in; otherwise, it uses your user-defined format plug-in.

Pragmas

none

Exception

If the video plug-in raises an exception when calling this method, then it raises a VIDEO_PLUGIN_EXCEPTION exception.

Example

Check property information for known video attributes:

```
DECLARE
  obj ORDSYS.ORDVideo;
  ctx RAW(4000) :=NULL;
BEGIN
  select vid into obj from TVID where N =1 ;
  if (obj.checkProperties(ctx)) then
  DBMS_OUTPUT.put_line('check Properties returned true');
else
```

```
  DBMS_OUTPUT.put_line('check Properties returned false');
  end if;
  EXCEPTION
    WHEN OTHERS THEN
     DBMS_OUTPUT.put_line('exception raised');
END;
/
```

getAttribute( ) Method

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.

Parameters

ctx
The format plug-in context information.

name
The name of the attribute.

Usage

The video data attributes are available from the header of the formatted video data.

If the format is set to NULL, then the getAttribute( ) method uses the default format plug-in; otherwise, it uses your user-defined format plug-in.

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 Section 2.3.13 for more information.

Pragmas

none

Exception

If the video plug-in raises an exception when calling this method, then it raises a VIDEO_PLUGIN_EXCEPTION exception.

Example

Return information for the specified video attribute for video data stored in the database:

```
DECLARE
  obj ORDSYS.ORDVideo;
  res VARCHAR2(4000);
  ctx RAW(4000) :=NULL;
BEGIN
  SELECT vid INTO obj FROM TVID  WHERE N=1;
  DBMS_OUTPUT.PUT_LINE('getting video duration');
  DBMS_OUTPUT.PUT_LINE('--------------------');
  res := obj.getAttribute(ctx,'video_duration');
  EXCEPTION
        WHEN ORDSYS.ORDSourceExceptions.METHOD_NOT_SUPPORTED THEN
         DBMS_OUTPUT.put_line('Source METHOD_NOT_SUPPORTED caught');
        WHEN ORDSYS.ORDSourceExceptions.SOURCE_PLUGIN_EXCEPTION THEN
         DBMS_OUTPUT.put_line('SOURCE PLUGIN EXCEPTION caught');
        WHEN ORDSYS.ORDVideoExceptions.METHOD_NOT_SUPPORTED THEN
         DBMS_OUTPUT.put_line('VIDEO METHOD_NOT_SUPPORTED EXCEPTION caught');
        WHEN ORDSYS.ORDVideoExceptions.VIDEO_PLUGIN_EXCEPTION THEN
         DBMS_OUTPUT.put_line('VIDEO PLUGIN EXCEPTION caught');
        WHEN OTHERS THEN
         DBMS_OUTPUT.PUT_LINE('EXCEPTION caught');
END;
/
```

getAllAttributes( ) Method

Format

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

Description

Returns a formatted string for convenient client access. For natively supported formats, the string includes the following list of audio data attributes separated by a comma (','): Format, FrameSize, FrameResolution, FrameRate, VideoDuration, and NumberOfFrames. For user-defined formats, the string is defined by the format plug-in.

Parameters

ctx
The format plug-in context information.

attributes
The attributes.

Usage

These video data attributes are available from the header of the formatted video data.

If the format is set to NULL, then the getAllAttributes( ) method uses the default format plug-in; otherwise, it uses your user-defined format plug-in.

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 Section 2.3.13 for more information.

Pragmas

none

Exception

If the video plug-in raises an exception when calling this method, then it raises either a METHOD_NOT_SUPPORTED exception or a VIDEO_PLUGIN_EXCEPTION exception.

Example

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

```
DECLARE
  obj ORDSYS.ORDVideo;
  tempLob   CLOB;
  ctx RAW(4000) :=NULL;
BEGIN

  SELECT vid INTO obj FROM TVID  WHERE N=1;
  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));
  EXCEPTION
  WHEN ORDSYS.ORDSourceExceptions.METHOD_NOT_SUPPORTED THEN
   DBMS_OUTPUT.put_line('Source METHOD_NOT_SUPPORTED caught');
  WHEN ORDSYS.ORDSourceExceptions.SOURCE_PLUGIN_EXCEPTION THEN
   DBMS_OUTPUT.put_line('SOURCE PLUGIN EXCEPTION caught');
  WHEN ORDSYS.ORDVideoExceptions.METHOD_NOT_SUPPORTED THEN
   DBMS_OUTPUT.put_line('VIDEO METHOD_NOT_SUPPORTED EXCEPTION caught');
  WHEN ORDSYS.ORDVideoExceptions.VIDEO_PLUGIN_EXCEPTION THEN
   DBMS_OUTPUT.put_line('VIDEO PLUGIN EXCEPTION caught');
  WHEN OTHERS THEN
   DBMS_OUTPUT.put_line('EXCEPTION CAUGHT');
END;
/
```

5.2.9 ORDVideo Methods Associated with Processing Video Data

This section presents reference information on the ORDVideo methods associated with processing video data.

processVideoCommand( ) Method

Format

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

Description

Allows you to send a command and related arguments to the format plug-in for processing. This method is supported only for user-defined format plug-ins.

Parameters

ctx
The format plug-in context information.

cmd
Any command recognized by the format plug-in.

arguments
The arguments of the command.

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

Usage

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 Section 2.3.13 for more information.

Pragmas

none

Exception

If the video plug-in raises an exception when calling this method, then it raises either a METHOD_NOT_SUPPORTED exception or a VIDEO_PLUGIN_EXCEPTION exception.

Example

Process a set of commands:

```
DECLARE
  obj ORDSYS.ORDVideo;
  res RAW(4000);
  result RAW(4000);
  command VARCHAR(4000);
  argList VARCHAR(4000);
  ctx RAW(4000) :=NULL;
BEGIN
select vid into obj from TVID where N =1 for UPDATE;
-- assign command
-- assign argList
res := obj.processVideoCommand (ctx, command, argList, result);
UPDATE TVID SET vid=obj WHERE N=1 ;
COMMIT;
EXCEPTION
        WHEN ORDSYS.ORDSourceExceptions.METHOD_NOT_SUPPORTED THEN
         DBMS_OUTPUT.put_line('Source METHOD_NOT_SUPPORTED caught');
        WHEN ORDSYS.ORDSourceExceptions.SOURCE_PLUGIN_EXCEPTION THEN
         DBMS_OUTPUT.put_line('SOURCE PLUGIN EXCEPTION caught');
        WHEN ORDSYS.ORDVideoExceptions.METHOD_NOT_SUPPORTED THEN
         DBMS_OUTPUT.put_line('VIDEO METHOD_NOT_SUPPORTED EXCEPTION caught');
        WHEN ORDSYS.ORDVideoExceptions.VIDEO_PLUGIN_EXCEPTION THEN
         DBMS_OUTPUT.put_line('VIDEO PLUGIN EXCEPTION caught');
        WHEN OTHERS THEN
         DBMS_OUTPUT.put_line('EXCEPTION caught');
END;
/
```

5.3 Packages or PL/SQL Plug-ins

This section presents reference information on the packages or PL/SQL plug-ins provided.

5.3.1 Video Format Packages

Use each of the following packages as a video format template in developing your own video format package. Note that these packages are just templates and that you must implement each of these packages yourself to support the methods described within each package.

- ORDPLUGINS.ORDX_DEFAULT_VIDEO
- ORDPLUGINS.ORDX_AVI_VIDEO
- ORDPLUGINS.ORDX_QT_VIDEO (see ORDPLUGINS.ORDX_MOOV_VIDEO)
- ORDPLUGINS.ORDX_MOOV_VIDEO
- ORDPLUGINS.ORDX_MPEG_VIDEO

5.3.2 ORDPLUGINS.ORDX_DEFAULT_VIDEO Package

Use the following provided ORDPLUGINS.ORDX_DEFAULT_VIDEO package as a guide in developing your own ORDPLUGINS.ORDX_<format>_VIDEO video format package.

```
CREATE OR REPLACE PACKAGE ORDX_DEFAULT_VIDEO
authid current_user
AS
--VIDEO ATTRIBUTES ACCESSORS
FUNCTION  getFormat(ctx IN OUT RAW, obj IN ORDSYS.ORDVideo)
          RETURN VARCHAR2;
FUNCTION  getAttribute(ctx IN OUT RAW,
                       obj IN ORDSYS.ORDVideo,
                       name IN VARCHAR2)
          RETURN VARCHAR2;
PROCEDURE getFrameSize(ctx IN OUT RAW,
                       obj IN ORDSYS.ORDVideo,
                       width OUT INTEGER,
                       height OUT INTEGER);
FUNCTION  getFrameResolution(ctx IN OUT RAW, obj IN ORDSYS.ORDVideo)
            RETURN INTEGER;
FUNCTION  getFrameRate(ctx IN OUT RAW, obj IN ORDSYS.ORDVideo)
          RETURN INTEGER;
FUNCTION  getVideoDuration(ctx IN OUT RAW, obj IN ORDSYS.ORDVideo)
```

```
          RETURN INTEGER;
FUNCTION  getNumberOfFrames(ctx IN OUT RAW, obj IN ORDSYS.ORDVideo)
          RETURN INTEGER;
FUNCTION  getCompressionType(ctx IN OUT RAW, obj IN ORDSYS.ORDVideo)
          RETURN VARCHAR2;
FUNCTION  getNumberOfColors(ctx IN OUT RAW, obj IN ORDSYS.ORDVideo)
          RETURN INTEGER;
FUNCTION  getBitRate(ctx IN OUT RAW, obj IN ORDSYS.ORDVideo)
          RETURN INTEGER;
PROCEDURE setProperties(ctx IN OUT RAW, obj IN OUT NOCOPY ORDSYS.ORDVideo);
FUNCTION checkProperties(ctx IN OUT RAW,obj IN ORDSYS.ORDVideo) RETURN NUMBER;

-- must return name=value; name=value; ...  pairs
PROCEDURE getAllAttributes(ctx IN OUT RAW,
                           obj IN ORDSYS.ORDVideo,
                           attributes IN OUT NOCOPY CLOB);
-- VIDEO PROCESSING METHODS
FUNCTION  processCommand(
                         ctx       IN OUT RAW,
                         obj       IN OUT NOCOPY ORDSYS.ORDVideo,
                         cmd       IN VARCHAR2,
                         arguments IN VARCHAR2,
                         result    OUT RAW)
         RETURN RAW;
PRAGMA RESTRICT_REFERENCES(getFormat, WNDS, WNPS, RNDS, RNPS);
PRAGMA RESTRICT_REFERENCES(getAttribute, WNDS, WNPS, RNDS, RNPS);
PRAGMA RESTRICT_REFERENCES(getFrameSize, WNDS, WNPS, RNDS, RNPS);
PRAGMA RESTRICT_REFERENCES(getFrameResolution, WNDS, WNPS, RNDS, RNPS);
PRAGMA RESTRICT_REFERENCES(getFrameRate, WNDS, WNPS, RNDS, RNPS);
PRAGMA RESTRICT_REFERENCES(getVideoDuration, WNDS, WNPS, RNDS, RNPS);
PRAGMA RESTRICT_REFERENCES(getNumberOfFrames, WNDS, WNPS, RNDS, RNPS);
PRAGMA RESTRICT_REFERENCES(getCompressionType, WNDS, WNPS, RNDS, RNPS);
PRAGMA RESTRICT_REFERENCES(getNumberOfColors, WNDS, WNPS, RNDS, RNPS);
PRAGMA RESTRICT_REFERENCES(getBitRate, WNDS, WNPS, RNDS, RNPS);

END;
/
```

Table 5–1 shows the methods supported in the ORDPLUGINS.ORDX_DEFAULT_VIDEO package and the exceptions raised if you call a method that is not supported.

Table 5–1 Methods Supported in the ORDPLUGINS.ORDX_DEFAULT_VIDEO Package

Name of Method	Level of Support
getFormat	Not supported - raises exceptions: METHOD_NOT_SUPPORTED and VIDEO_PLUGIN_EXCEPTION
getAttribute	Not supported - raises exceptions: METHOD_NOT_SUPPORTED and VIDEO_PLUGIN_EXCEPTION
getFrameSize	Not supported - raises exceptions: METHOD_NOT_SUPPORTED and VIDEO_PLUGIN_EXCEPTION
getFrameResolution	Not supported - raises exceptions: METHOD_NOT_SUPPORTED and VIDEO_PLUGIN_EXCEPTION
getFrameRate	Not supported - raises exceptions: METHOD_NOT_SUPPORTED and VIDEO_PLUGIN_EXCEPTION
getVideoDuration	Not supported - raises exceptions: METHOD_NOT_SUPPORTED and VIDEO_PLUGIN_EXCEPTION
getNumberOfFrames	Not supported - raises exceptions: METHOD_NOT_SUPPORTED and VIDEO_PLUGIN_EXCEPTION
getCompressionType	Not supported - raises exceptions: METHOD_NOT_SUPPORTED and VIDEO_PLUGIN_EXCEPTION
getNumberOfColors	Not supported - raises exceptions: METHOD_NOT_SUPPORTED and VIDEO_PLUGIN_EXCEPTION
getBitRate	Not supported - raises exceptions: METHOD_NOT_SUPPORTED and VIDEO_PLUGIN_EXCEPTION
setProperties	Not supported - raises exceptions: METHOD_NOT_SUPPORTED and VIDEO_PLUGIN_EXCEPTION
checkProperties	Not supported - raises exceptions: METHOD_NOT_SUPPORTED and VIDEO_PLUGIN_EXCEPTION
getAllAttributes	Not supported - raises exceptions: METHOD_NOT_SUPPORTED and VIDEO_PLUGIN_EXCEPTION
processCommand	Not supported - raises exceptions: METHOD_NOT_SUPPORTED and VIDEO_PLUGIN_EXCEPTION

5.3.3 ORDPLUGINS.ORDX_AVI_VIDEO Package

The ORDPLUGINS.ORDX_AVI_VIDEO package is provided and is used for the AVI video format specification.

```
CREATE OR REPLACE PACKAGE ORDX_AVI_VIDEO
authid current_user
AS
```

```
--VIDEO ATTRIBUTES ACCESSORS
FUNCTION  getFormat(ctx IN OUT RAW, obj IN ORDSYS.ORDVideo)
          RETURN VARCHAR2;
FUNCTION  getAttribute(ctx IN OUT RAW,
                       obj IN ORDSYS.ORDVideo,
                       name IN VARCHAR2)
          RETURN VARCHAR2;
PROCEDURE getFrameSize(ctx IN OUT RAW,
                       obj IN ORDSYS.ORDVideo,
                       width OUT INTEGER,
                       height OUT INTEGER);
FUNCTION  getFrameResolution(ctx IN OUT RAW, obj IN ORDSYS.ORDVideo)
          RETURN INTEGER;
FUNCTION  getFrameRate(ctx IN OUT RAW, obj IN ORDSYS.ORDVideo)
          RETURN INTEGER;
FUNCTION  getVideoDuration(ctx IN OUT RAW, obj IN ORDSYS.ORDVideo)
          RETURN INTEGER;
FUNCTION  getNumberOfFrames(ctx IN OUT RAW, obj IN ORDSYS.ORDVideo)
          RETURN INTEGER;
FUNCTION  getCompressionType(ctx IN OUT RAW, obj IN ORDSYS.ORDVideo)
          RETURN VARCHAR2;
FUNCTION  getNumberOfColors(ctx IN OUT RAW, obj IN ORDSYS.ORDVideo)
          RETURN INTEGER;
FUNCTION  getBitRate(ctx IN OUT RAW, obj IN ORDSYS.ORDVideo)
          RETURN INTEGER;
PROCEDURE setProperties(ctx IN OUT RAW, obj IN OUT NOCOPY ORDSYS.ORDVideo);
FUNCTION checkProperties(ctx IN OUT RAW,obj IN ORDSYS.ORDVideo) RETURN NUMBER;

-- must return name=value; name=value; ...  pairs
PROCEDURE getAllAttributes(ctx IN OUT RAW,
                           obj IN ORDSYS.ORDVideo,
                           attributes IN OUT NOCOPY CLOB);
-- VIDEO PROCESSING METHODS
FUNCTION  processCommand(
                         ctx       IN OUT RAW,
                         obj       IN OUT NOCOPY ORDSYS.ORDVideo,
                         cmd       IN VARCHAR2,
                         arguments IN VARCHAR2,
                         result    OUT RAW)
          RETURN RAW;
PRAGMA RESTRICT_REFERENCES(getFormat, WNDS, WNPS, RNDS, RNPS);
PRAGMA RESTRICT_REFERENCES(getAttribute, WNDS, WNPS, RNDS, RNPS);
PRAGMA RESTRICT_REFERENCES(getFrameSize, WNDS, WNPS, RNDS, RNPS);
PRAGMA RESTRICT_REFERENCES(getFrameResolution, WNDS, WNPS, RNDS, RNPS);
PRAGMA RESTRICT_REFERENCES(getFrameRate, WNDS, WNPS, RNDS, RNPS);
```

```
PRAGMA RESTRICT_REFERENCES(getVideoDuration, WNDS, WNPS, RNDS, RNPS);
PRAGMA RESTRICT_REFERENCES(getNumberOfFrames, WNDS, WNPS, RNDS, RNPS);
PRAGMA RESTRICT_REFERENCES(getCompressionType, WNDS, WNPS, RNDS, RNPS);
PRAGMA RESTRICT_REFERENCES(getNumberOfColors, WNDS, WNPS, RNDS, RNPS);
PRAGMA RESTRICT_REFERENCES(getBitRate, WNDS, WNPS, RNDS, RNPS);

END;
/
```

Table 5–2 shows the methods supported in the ORDPLUGINS.ORDX_AVI_VIDEO package and the exceptions raised if you call a method that is not supported.

Table 5–2 Methods Supported in the ORDPLUGINS.ORDX_AVI_VIDEO Package

Name of Method	Level of Support
getFormat	Not supported - raises exceptions: METHOD_NOT_SUPPORTED and VIDEO_PLUGIN_EXCEPTION
getAttribute	Not supported - raises exceptions: METHOD_NOT_SUPPORTED and VIDEO_PLUGIN_EXCEPTION
getFrameSize	Not supported - raises exceptions: METHOD_NOT_SUPPORTED and VIDEO_PLUGIN_EXCEPTION
getFrameResolution	Not supported - raises exceptions: METHOD_NOT_SUPPORTED and VIDEO_PLUGIN_EXCEPTION
getFrameRate	Not supported - raises exceptions: METHOD_NOT_SUPPORTED and VIDEO_PLUGIN_EXCEPTION
getVideoDuration	Not supported - raises exceptions: METHOD_NOT_SUPPORTED and VIDEO_PLUGIN_EXCEPTION
getNumberOfFrames	Not supported - raises exceptions: METHOD_NOT_SUPPORTED and VIDEO_PLUGIN_EXCEPTION
getCompressionType	Not supported - raises exceptions: METHOD_NOT_SUPPORTED and VIDEO_PLUGIN_EXCEPTION
getNumberOfColors	Not supported - raises exceptions: METHOD_NOT_SUPPORTED and VIDEO_PLUGIN_EXCEPTION
getBitRate	Not supported - raises exceptions: METHOD_NOT_SUPPORTED and VIDEO_PLUGIN_EXCEPTION
setProperties	Not supported - raises exceptions: METHOD_NOT_SUPPORTED and VIDEO_PLUGIN_EXCEPTION
checkProperties	Not supported - raises exceptions: METHOD_NOT_SUPPORTED and VIDEO_PLUGIN_EXCEPTION

Table 5–2 Methods Supported in the ORDPLUGINS.ORDX_AVI_VIDEO Package (Cont.)

Name of Method	Level of Support
getAllAttributes	Not supported - raises exceptions: METHOD_NOT_SUPPORTED and VIDEO_PLUGIN_EXCEPTION
processCommand	Not supported - raises exceptions METHOD_NOT_SUPPORTED and VIDEO_PLUGIN_EXCEPTION

5.3.4 ORDPLUGINS.ORDX_MOOV_VIDEO Package

The ORDPLUGINS.ORDX_MOOV_VIDEO package is provided and is used for the MOOV video format specification.

```
CREATE OR REPLACE PACKAGE ORDX_MOOV_VIDEO
authid current_user
AS
--VIDEO ATTRIBUTES ACCESSORS
FUNCTION  getFormat(ctx IN OUT RAW, obj IN ORDSYS.ORDVideo)
          RETURN VARCHAR2;
FUNCTION  getAttribute(ctx IN OUT RAW,
                       obj IN ORDSYS.ORDVideo,
                       name IN VARCHAR2)
          RETURN VARCHAR2;
PROCEDURE getFrameSize(ctx IN OUT RAW,
                       obj IN ORDSYS.ORDVideo,
                       width OUT INTEGER,
                       height OUT INTEGER);
FUNCTION  getFrameResolution(ctx IN OUT RAW, obj IN ORDSYS.ORDVideo)
          RETURN INTEGER;
FUNCTION  getFrameRate(ctx IN OUT RAW, obj IN ORDSYS.ORDVideo)
          RETURN INTEGER;
FUNCTION  getVideoDuration(ctx IN OUT RAW, obj IN ORDSYS.ORDVideo)
          RETURN INTEGER;
FUNCTION  getNumberOfFrames(ctx IN OUT RAW, obj IN ORDSYS.ORDVideo)
          RETURN INTEGER;
FUNCTION  getCompressionType(ctx IN OUT RAW, obj IN ORDSYS.ORDVideo)
          RETURN VARCHAR2;
FUNCTION  getNumberOfColors(ctx IN OUT RAW, obj IN ORDSYS.ORDVideo)
          RETURN INTEGER;
FUNCTION  getBitRate(ctx IN OUT RAW, obj IN ORDSYS.ORDVideo)
          RETURN INTEGER;
PROCEDURE setProperties(ctx IN OUT RAW, obj IN OUT NOCOPY ORDSYS.ORDVideo);
FUNCTION checkProperties(ctx IN OUT RAW,obj IN ORDSYS.ORDVideo) RETURN NUMBER;

-- must return name=value; name=value; ...  pairs
```

```
PROCEDURE getAllAttributes(ctx IN OUT RAW,
                           obj IN ORDSYS.ORDVideo,
                           attributes IN OUT NOCOPY CLOB);
-- VIDEO PROCESSING METHODS
FUNCTION  processCommand(
                         ctx       IN OUT RAW,
                         obj       IN OUT NOCOPY ORDSYS.ORDVideo,
                         cmd       IN VARCHAR2,
                         arguments IN VARCHAR2,
                         result    OUT RAW)
          RETURN RAW;
PRAGMA RESTRICT_REFERENCES(getFormat, WNDS, WNPS, RNDS, RNPS);
PRAGMA RESTRICT_REFERENCES(getAttribute, WNDS, WNPS, RNDS, RNPS);
PRAGMA RESTRICT_REFERENCES(getFrameSize, WNDS, WNPS, RNDS, RNPS);
PRAGMA RESTRICT_REFERENCES(getFrameResolution, WNDS, WNPS, RNDS, RNPS);
PRAGMA RESTRICT_REFERENCES(getFrameRate, WNDS, WNPS, RNDS, RNPS);
PRAGMA RESTRICT_REFERENCES(getVideoDuration, WNDS, WNPS, RNDS, RNPS);
PRAGMA RESTRICT_REFERENCES(getNumberOfFrames, WNDS, WNPS, RNDS, RNPS);
PRAGMA RESTRICT_REFERENCES(getCompressionType, WNDS, WNPS, RNDS, RNPS);
PRAGMA RESTRICT_REFERENCES(getNumberOfColors, WNDS, WNPS, RNDS, RNPS);
PRAGMA RESTRICT_REFERENCES(getBitRate, WNDS, WNPS, RNDS, RNPS);

END;
/
```

Table 5–3 shows the methods supported in the ORDPLUGINS.ORDX_MOOV_VIDEO package and the exceptions raised if you call a method that is not supported.

Table 5–3 Methods Supported in the ORDPLUGINS.ORDX_MOOV_VIDEO Package

Name of Method	Level of Support
getFormat	Not supported - raises exceptions: METHOD_NOT_SUPPORTED and VIDEO_PLUGIN_EXCEPTION
getAttribute	Not supported - raises exceptions: METHOD_NOT_SUPPORTED and VIDEO_PLUGIN_EXCEPTION
getFrameSize	Not supported - raises exceptions: METHOD_NOT_SUPPORTED and VIDEO_PLUGIN_EXCEPTION
getFrameResolution	Not supported - raises exceptions: METHOD_NOT_SUPPORTED and VIDEO_PLUGIN_EXCEPTION
getFrameRate	Not supported - raises exceptions: METHOD_NOT_SUPPORTED and VIDEO_PLUGIN_EXCEPTION

Table 5–3 Methods Supported in the ORDPLUGINS.ORDX_MOOV_VIDEO Package (Cont.)

Name of Method	Level of Support
getVideoDuration	Not supported - raises exceptions: METHOD_NOT_SUPPORTED and VIDEO_PLUGIN_EXCEPTION
getNumberOfFrames	Not supported - raises exceptions: METHOD_NOT_SUPPORTED and VIDEO_PLUGIN_EXCEPTION
getCompressionType	Not supported - raises exceptions: METHOD_NOT_SUPPORTED and VIDEO_PLUGIN_EXCEPTION
getNumberOfColors	Not supported - raises exceptions: METHOD_NOT_SUPPORTED and VIDEO_PLUGIN_EXCEPTION
getBitRate	Not supported - raises exceptions: METHOD_NOT_SUPPORTED and VIDEO_PLUGIN_EXCEPTION
setProperties	Not supported - raises exceptions: METHOD_NOT_SUPPORTED and VIDEO_PLUGIN_EXCEPTION
checkProperties	Not supported - raises exceptions: METHOD_NOT_SUPPORTED and VIDEO_PLUGIN_EXCEPTION
getAllAttributes	Not supported - raises exceptions: METHOD_NOT_SUPPORTED and VIDEO_PLUGIN_EXCEPTION
processCommand	Not supported - raises exceptions: METHOD_NOT_SUPPORTED and VIDEO_PLUGIN_EXCEPTION

5.3.5 ORDPLUGINS.ORDX_MPEG_VIDEO Package

The ORDPLUGINS.ORDX_MPEG_VIDEO package is provided and is used for the MPEG video format specification.

```
CREATE OR REPLACE PACKAGE ORDX_MPEG_VIDEO
authid current_user
AS
--VIDEO ATTRIBUTES ACCESSORS
FUNCTION  getFormat(ctx IN OUT RAW, obj IN ORDSYS.ORDVideo)
          RETURN VARCHAR2;
FUNCTION  getAttribute(ctx IN OUT RAW,
                       obj IN ORDSYS.ORDVideo,
                       name IN VARCHAR2)
          RETURN VARCHAR2;
PROCEDURE getFrameSize(ctx IN OUT RAW,
                       obj IN ORDSYS.ORDVideo,
                       width OUT INTEGER,
                       height OUT INTEGER);
FUNCTION  getFrameResolution(ctx IN OUT RAW, obj IN ORDSYS.ORDVideo)
          RETURN INTEGER;
```

```
FUNCTION  getFrameRate(ctx IN OUT RAW, obj IN ORDSYS.ORDVideo)
          RETURN INTEGER;
FUNCTION  getVideoDuration(ctx IN OUT RAW, obj IN ORDSYS.ORDVideo)
          RETURN INTEGER;
FUNCTION  getNumberOfFrames(ctx IN OUT RAW, obj IN ORDSYS.ORDVideo)
          RETURN INTEGER;
FUNCTION  getCompressionType(ctx IN OUT RAW, obj IN ORDSYS.ORDVideo)
          RETURN VARCHAR2;
FUNCTION  getNumberOfColors(ctx IN OUT RAW, obj IN ORDSYS.ORDVideo)
          RETURN INTEGER;
FUNCTION  getBitRate(ctx IN OUT RAW, obj IN ORDSYS.ORDVideo)
          RETURN INTEGER;
PROCEDURE setProperties(ctx IN OUT RAW, obj IN OUT NOCOPY ORDSYS.ORDVideo);
FUNCTION checkProperties(ctx IN OUT RAW,obj IN ORDSYS.ORDVideo) RETURN NUMBER;

-- must return name=value; name=value; ...  pairs
PROCEDURE getAllAttributes(ctx IN OUT RAW,
                           obj IN ORDSYS.ORDVideo,
                           attributes IN OUT NOCOPY CLOB);
-- VIDEO PROCESSING METHODS
FUNCTION  processCommand(
                         ctx       IN OUT RAW,
                         obj       IN OUT NOCOPY ORDSYS.ORDVideo,
                         cmd       IN VARCHAR2,
                         arguments IN VARCHAR2,
                         result    OUT RAW)
          RETURN RAW;
PRAGMA RESTRICT_REFERENCES(getFormat, WNDS, WNPS, RNDS, RNPS);
PRAGMA RESTRICT_REFERENCES(getAttribute, WNDS, WNPS, RNDS, RNPS);
PRAGMA RESTRICT_REFERENCES(getFrameSize, WNDS, WNPS, RNDS, RNPS);
PRAGMA RESTRICT_REFERENCES(getFrameResolution, WNDS, WNPS, RNDS, RNPS);
PRAGMA RESTRICT_REFERENCES(getFrameRate, WNDS, WNPS, RNDS, RNPS);
PRAGMA RESTRICT_REFERENCES(getVideoDuration, WNDS, WNPS, RNDS, RNPS);
PRAGMA RESTRICT_REFERENCES(getNumberOfFrames, WNDS, WNPS, RNDS, RNPS);
PRAGMA RESTRICT_REFERENCES(getCompressionType, WNDS, WNPS, RNDS, RNPS);
PRAGMA RESTRICT_REFERENCES(getNumberOfColors, WNDS, WNPS, RNDS, RNPS);
PRAGMA RESTRICT_REFERENCES(getBitRate, WNDS, WNPS, RNDS, RNPS);

END;
/
```

Table 5–4 shows the methods supported in the ORDPLUGINS.ORDX_MPEG_VIDEO package and the exceptions raised if you call a method that is not supported.

Table 5–4 Methods Supported in the ORDPLUGINS.ORDX_MPEG_VIDEO Package

Name of Method	Level of Support
getFormat	Not supported - raises exceptions: METHOD_NOT_SUPPORTED and VIDEO_PLUGIN_EXCEPTION
getAttribute	Not supported - raises exceptions: METHOD_NOT_SUPPORTED and VIDEO_PLUGIN_EXCEPTION
getFrameSize	Not supported - raises exceptions: METHOD_NOT_SUPPORTED and VIDEO_PLUGIN_EXCEPTION
getFrameResolution	Not supported - raises exceptions: METHOD_NOT_SUPPORTED and VIDEO_PLUGIN_EXCEPTION
getFrameRate	Not supported - raises exceptions: METHOD_NOT_SUPPORTED and VIDEO_PLUGIN_EXCEPTION
getVideoDuration	Not supported - raises exceptions: METHOD_NOT_SUPPORTED and VIDEO_PLUGIN_EXCEPTION
getNumberOfFrames	Not supported - raises exceptions: METHOD_NOT_SUPPORTED and VIDEO_PLUGIN_EXCEPTION
getCompressionType	Not supported - raises exceptions: METHOD_NOT_SUPPORTED and VIDEO_PLUGIN_EXCEPTION
getNumberOfColors	Not supported - raises exceptions: METHOD_NOT_SUPPORTED and VIDEO_PLUGIN_EXCEPTION
getBitRate	Not supported - raises exceptions: METHOD_NOT_SUPPORTED and VIDEO_PLUGIN_EXCEPTION
setProperties	Not supported - raises exceptions: METHOD_NOT_SUPPORTED and VIDEO_PLUGIN_EXCEPTION
checkProperties	Not supported - raises exceptions: METHOD_NOT_SUPPORTED and VIDEO_PLUGIN_EXCEPTION
getAllAttributes	Not supported - raises exceptions: METHOD_NOT_SUPPORTED and VIDEO_PLUGIN_EXCEPTION
processCommand	Not supported - raises exceptions: METHOD_NOT_SUPPORTED and VIDEO_PLUGIN_EXCEPTION

5.3.6 Extending *inter*Media to Support a New Video Data Format

Extending *inter*Media to support a new video data format consists of three steps:

1. Design your new video data format.
2. Implement your new video data format and name it, for example, ORDX_MY_VIDEO.SQL.

3. Install your new ORDX_MY_VIDEO.SQL plug-in in the ORDPLUGINS schema.
4. Grant EXECUTE privileges on your new plug-in, for example, ORDX_MY_VIDEO.SQL plug-in, to PUBLIC.

Section 2.3.12 briefly describes how to extend *inter*Media to support a new video data format and describes the interface. A package body listing is provided in Example 5–1 to assist you in this operation. Add your variables to the places that say "--Your variables go here" and add your code to the places that say "--Your code goes here".

See Section E.3 for more information on installing your own video format plug-in and running the sample scripts provided.

Example 5–1 Package Body Listing for Extending Support to a New Video Data Format

```
CREATE OR REPLACE PACKAGE BODY ORDX_MY_VIDEO
AS
  --VIDEO ATTRIBUTES ACCESSORS
  FUNCTION  getFormat(ctx IN OUT RAW, obj IN ORDSYS.ORDVideo)
  RETURN VARCHAR2
  IS
--Your variables go here
  BEGIN
--Your code goes here
  END;
  FUNCTION  getAttribute(ctx IN OUT RAW,
                         obj IN ORDSYS.ORDVideo,
                         name IN VARCHAR2)
  RETURN VARCHAR2
  IS
--Your variables go here
  BEGIN
--Your code goes here
  END;
  PROCEDURE getFrameSize(ctx IN OUT RAW,
                         obj IN ORDSYS.ORDVideo,
                         width OUT INTEGER,
                         height OUT INTEGER)
  IS
--Your variables go here
  BEGIN
--Your code goes here
  END;
  FUNCTION  getFrameResolution(ctx IN OUT RAW, obj IN ORDSYS.ORDVideo)
```

```
  RETURN INTEGER
  IS
--Your variables go here
  BEGIN
--Your code goes here
  END;
  FUNCTION  getFrameRate(ctx IN OUT RAW, obj IN ORDSYS.ORDVideo)
  RETURN INTEGER
  IS
--Your variables go here
  BEGIN
--Your code goes here
  END;
  FUNCTION  getVideoDuration(ctx IN OUT RAW, obj IN ORDSYS.ORDVideo)
  RETURN INTEGER
  IS
--Your variables go here
  BEGIN
--Your code goes here
  END;
  FUNCTION  getNumberOfFrames(ctx IN OUT RAW, obj IN ORDSYS.ORDVideo)
  RETURN INTEGER
  IS
--Your variables go here
  BEGIN
--Your code goes here
  END;
  FUNCTION  getCompressionType(ctx IN OUT RAW, obj IN ORDSYS.ORDVideo)
  RETURN VARCHAR2
  IS
--Your variables go here
  BEGIN
--Your code goes here
  END;
  FUNCTION  getNumberOfColors(ctx IN OUT RAW, obj IN ORDSYS.ORDVideo)
  RETURN INTEGER
  IS
--Your variables go here
  BEGIN
--Your code goes here
  END;
  FUNCTION  getBitRate(ctx IN OUT RAW, obj IN ORDSYS.ORDVideo)
  RETURN INTEGER
  IS
--Your variables go here
```

```
  BEGIN
--Your code goes here
  END;
  PROCEDURE setProperties(ctx IN OUT RAW, obj IN OUT NOCOPY ORDSYS.ORDVideo)
  IS
--Your variables go here
  BEGIN
--Your code goes here
  END;
  FUNCTION checkProperties(ctx IN OUT RAW, obj IN ORDSYS.ORDVideo) RETURN NUMBER
  IS
  IS
--Your variables go here
  BEGIN
--Your code goes here
  END;
  PROCEDURE getAllAttributes(ctx IN OUT RAW,
                             obj IN ORDSYS.ORDVideo,
                             attributes IN OUT NOCOPY CLOB)
  IS
--Your variables go here
  BEGIN
--Your code goes here
  END;
  -- VIDEO PROCESSING METHODS
  FUNCTION  processCommand(
                                ctx       IN OUT RAW,
                                obj       IN OUT NOCOPY ORDSYS.ORDVideo,
                                cmd       IN VARCHAR2,
                                arguments IN VARCHAR2,
                                result OUT RAW)
  RETURN RAW
  IS
--Your variables go here
  BEGIN
--Your code goes here
  END;
END;
/
show errors;
```

6

ORDSource Reference Information

Oracle8*i* *inter*Media contains the following information about the ORDSource type:

- Object type -- see Section 6.1.
- Methods -- see Section 6.2.
- Packages or PL/SQL plug-ins -- see Section 6.3.

This object is used only by other Oracle8*i* *inter*Media objects. You can use this as an embedded object to implement source mechanisms for your own objects.

The examples in this chapter assume that the test source table TS has been created and filled with data. This table was created using the SQL statements described in Section 6.2.1.

Methods invoked at the ORDSource level that are handed off to the source plug-in for processing have ctx (RAW(4000)) 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 should invoke the close( ) method.

Methods invoked from a source plug-in call have the first argument as obj (ORDSource) and the second argument as ctx (RAW(4000)).

Note: In the current release, not all source plug-ins will use the ctx argument, but if you code as previously described, your application should work with any current or future source plug-in.

The ORDSource object does not attempt to maintain consistency, for example, with local and upDateTime attributes. It is up to you to maintain consistency. ORDAu-

dio, ORDImage, and ORDVideo objects all maintain consistency of their included ORDSource object.

6.1 Object Types

Oracle8*i* *inter*Media provides the ORDSource object type, which supports access to a variety of sources of multimedia data.

ORDSource Object Type

The ORDSource object type supports access to data sources locally in a BLOB within an Oracle database, externally from a BFILE on a local file system, externally from a URL on an HTTP server (within the firewall), or externally from a user-defined source on another server. This object type is defined as follows:

```
CREATE OR REPLACE TYPE ORDsource
AS OBJECT
(
  -- ATTRIBUTES
localData           BLOB,
srcType             VARCHAR2(4000),
srcLocation         VARCHAR2(4000),
srcName             VARCHAR2(4000),
updateTime          DATE,
local               NUMBER,
  -- METHODS
-- Methods associated with the local attribute
MEMBER PROCEDURE setLocal,
MEMBER PROCEDURE clearLocal,
MEMBER FUNCTION isLocal RETURN BOOLEAN,
PRAGMA RESTRICT_REFERENCES(isLocal, WNDS, WNPS, RNDS, RNPS),
-- Methods associated with the updateTime attribute
MEMBER FUNCTION getUpdateTime RETURN DATE,
PRAGMA RESTRICT_REFERENCES(getUpdateTime, WNDS, WNPS, RNDS, RNPS),
MEMBER PROCEDURE setUpdateTime(current_time DATE),
-- Methods associated with the source information
MEMBER PROCEDURE setSourceInformation(
                                    source_type     IN VARCHAR2,
                                    source_location IN VARCHAR2,
                                    source_name     IN VARCHAR2),
MEMBER FUNCTION getSourceInformation RETURN VARCHAR2,
PRAGMA RESTRICT_REFERENCES(getSourceInformation, WNDS, WNPS, RNDS, RNPS),

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

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

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

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

-- Methods associated with source import/export operations
MEMBER PROCEDURE import(
                       ctx      IN OUT RAW,
                       dlob     IN OUT NOCOPY BLOB,
                       mimetype OUT VARCHAR2,
                       format   OUT VARCHAR2),
MEMBER PROCEDURE importFrom(
                           ctx             IN OUT RAW,
                           dlob            IN OUT NOCOPY BLOB,
                           mimetype        OUT VARCHAR2,
                           format          OUT VARCHAR2,
                           source_type     IN VARCHAR2,
                           source_location IN VARCHAR2,
                           source_name     IN VARCHAR2),
MEMBER PROCEDURE export(
                       ctx             IN OUT RAW,
                       source_type     IN VARCHAR2,
                       source_location IN VARCHAR2,
                       source_name     IN VARCHAR2),
-- Methods associated with source content-related operations
MEMBER FUNCTION getContentLength(ctx IN OUT RAW) RETURN INTEGER,
PRAGMA RESTRICT_REFERENCES(getContentLength, WNDS, WNPS, RNDS, RNPS),

MEMBER FUNCTION getSourceAddress(ctx IN OUT RAW,
                                 userData IN VARCHAR2)
                RETURN VARCHAR2,
PRAGMA RESTRICT_REFERENCES(getSourceAddress, WNDS, WNPS, RNDS, RNPS),

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

MEMBER PROCEDURE getContentInTempLob(
                                    ctx      IN OUT RAW,
                                    tempLob  IN OUT NOCOPY BLOB,
                                    mimetype OUT VARCHAR2,
                                    format   OUT VARCHAR2,
                                    duration IN PLS_INTEGER := 10,
                                    cache    IN BOOLEAN := TRUE),
MEMBER PROCEDURE deleteLocalContent,

-- Methods associated with source access methods
```

```
MEMBER FUNCTION open(userArg IN RAW, ctx OUT RAW) RETURN INTEGER,
MEMBER FUNCTION close(ctx IN OUT RAW) RETURN INTEGER,
MEMBER FUNCTION trim(ctx     IN OUT RAW,
                     newlen  IN INTEGER) RETURN INTEGER,

-- Methods associated with content read/write operations
MEMBER PROCEDURE read(
                      ctx      IN OUT RAW,
                      startPos IN INTEGER,
                      numBytes IN OUT INTEGER,
                      buffer   OUT RAW),
MEMBER PROCEDURE write(
                       ctx      IN OUT RAW,
                       startPos IN INTEGER,
                       numBytes IN OUT INTEGER,
                       buffer   IN RAW),
-- Methods associated with any commands to be sent to the external source
MEMBER FUNCTION processCommand(
                               ctx     IN OUT RAW,
                               command IN VARCHAR2,
                               arglist IN VARCHAR2,
                               result  OUT RAW)
               RETURN RAW
);
```

where:

- localData: contains the locally stored multimedia data stored as a BLOB within the object. Up to 4 gigabytes of data can be stored as a BLOB within an Oracle database and is protected by the Oracle security and transaction environment.
- srcType: identifies the data source type. Supported values source types are:

srcType	Source Type
"FILE"	A BFILE on a local file system
"HTTP"	An HTTP server
"<name>"	User-defined

> **Note:** The keyword 'FILE' for the plug-in is a reserved word for the BFILE source provided by Oracle Corporation. To implement for your own file plug-in, select a different name, for example, 'MYFILE'.

- srcLocation: identifies the place where data can be found based on the srcType value. Valid srcLocation values for corresponding srcType values are:

srcType	Location Value
"FILE"	<DIR> or name of the directory object
"HTTP"	<SourceBase> or URL needed to find the base directory
"<name>"	<iden> or identifier string required to access a user-defined source

- srcName: identifies the data object name. Valid srcName values for corresponding srcType values are:

srcType	Name Value
"FILE"	<FILE> or name of the file
"HTTP"	<Source> or name of the object
<name>	<object name> or name of the object

- updateTime: the time at which the data was last updated.
- local: a flag to determine whether or not the data is local:

  1 means the data is in the BLOB.

  0 means the data is in external sources.

  NULL, which may be a default state when you first insert an empty row, is assumed to mean data is local.

6.2 Methods

This section presents ORDSource reference information on the Oracle8*i* *inter*Media methods provided for source data manipulation. These methods are described in the following groupings:

ORDSource Methods Associated with the local Attribute

- setLocal: sets the flag value for the local attribute to "1", meaning that the source of the data is local.
- clearLocal: resets the flag value for the local attribute to "0", meaning that the source of the data is external.
- isLocal: returns TRUE to indicate that the source of the data is local or in the BLOB, or FALSE, meaning the data is in an external source. The value of the local attribute is used to determine the return value.

ORDSource Methods Associated with the updateTime Attribute

- getUpdateTime: returns the value of the updateTime attribute.
- setUpdateTime: sets the value of the updateTime attribute to the specified time provided in the argument.

ORDSource Methods Associated with the srcType, srcLocation, and srcName Attributes

- setSourceInformation( ): sets or alters information about the source of the data.
- getSourceInformation: returns a formatted string containing complete information about the data source formatted as a URL.
- getSourceType: returns the external source type of the data.
- getSourceLocation: returns the external source location of the data.
- getSourceName: returns the external source name of the data.
- getBFile: returns the external content as a BFILE, if srcType is of type FILE.

ORDSource Methods Associated with import and export Operations

- import( ): transfers data from an external data source (specified by calling setSourceInformation( )) to the local source (localData) within an Oracle database.
- importFrom( ): transfers data from the specified external data source (source, location, name) to the local source (localData) within an Oracle database.

- export( ): transfers data from a local source (localData) within an Oracle database to the specified external data source. This method is supported only for user-defined sources.

ORDSource Methods Associated with the localData Attribute

- getContentLength( ): returns the length of the data source (as number of bytes).
- getSourceAddress( ): returns the address of the data source.
- getLocalContent: returns the handle to the BLOB used to store contents locally.
- getContentInTempLob( ): returns content into a temporary LOB.
- deleteLocalContent: deletes the content of the local BLOB.

ORDSource Methods Associated with Access Operations

- open( ): opens a data source.
- close( ): closes a data source.
- trim( ): trims a data source.

ORDSource Methods Associated with Source Read/Write Operations

- read( ): reads a buffer of n bytes from a source beginning at a start position.
- write( ): writes a buffer of n bytes to a source beginning at a start position.

ORDSource Methods Associated with Processing Commands to the External Source

- processCommand( ): process as any command to the external source. This method is supported only for user-defined sources.

For more information on object types and methods, see the *Oracle8i Concepts* manual.

6.2.1 Example Table Definitions

The methods described in this reference chapter show examples based on a test source table TS. Refer to the TS table definition that follows when reading through the examples in Section 6.2.2 through Section 6.2.9:

TS Table Definition

```
CREATE TABLE TS(n NUMBER, s ORDSYS.ORDSOURCE);
```

```
INSERT INTO TS VALUES(1, ORDSYS.ORDSOURCE(EMPTY_BLOB(), NULL, NULL, NULL,
SYSDATE, NULL));
INSERT INTO TS VALUES(2, ORDSYS.ORDSOURCE(EMPTY_BLOB(), NULL, NULL, NULL,
SYSDATE, NULL));
INSERT INTO TS VALUES(3, ORDSYS.ORDSOURCE(EMPTY_BLOB(), NULL, NULL, NULL,
SYSDATE, NULL));
INSERT INTO TS VALUES(4, ORDSYS.ORDSOURCE(EMPTY_BLOB(), NULL, NULL, NULL,
SYSDATE, NULL));
```

6.2.2 ORDSource Methods Associated with the local Attribute

This section presents reference information on the ORDSource methods associated with the local attribute.

setLocal Method

Format

```
setLocal;
```

Description

Sets the local attribute to indicate that the data is stored in a BLOB within Oracle8*i*.

Parameters

none

Usage

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

Pragmas

none

Exception

none

Example

Set the flag to local for the data:

```
DECLARE
 SRC ORDSYS.ORDSource;
BEGIN
 SELECT S INTO SRC FROM TS WHERE N = 1 FOR UPDATE;
 SRC.setLocal;
 UPDATE TS SET S=SRC WHERE N = 1;
 COMMIT;
END;
/
```

clearLocal Method

Format

```
clearLocal;
```

Description

Resets the flag value from local, meaning the source of the data is stored locally in a BLOB in Oracle8*i*, to nonlocal meaning the source of the data is stored externally.

Parameters

none

Usage

This method sets the local attribute to a 0, meaning the data is stored externally or outside of Oracle8*i*.

Pragmas

none

Exception

none

Example

Clear the value of the local flag for the data:

```
DECLARE
 SRC ORDSYS.ORDSource;
BEGIN
 SELECT S INTO SRC FROM TS WHERE N = 1 FOR UPDATE;
 SRC.clearLocal;
 UPDATE TS SET S=SRC WHERE N = 1;
 COMMIT;
END;
/
```

isLocal Method

Format

```
isLocal RETURN BOOLEAN;
```

Description

Returns TRUE if the data is stored locally in a BLOB in Oracle8*i* or FALSE if the data is stored externally.

Parameters

none

Usage

If the local attribute is set to1 or NULL, this method returns TRUE, otherwise this method returns FALSE.

Pragmas

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

Exception

none

Example

Determine whether or not the data is local:

```
DECLARE
 SRC ORDSYS.ORDSource;
BEGIN
 SELECT S INTO SRC FROM TS WHERE N = 1 ;
 if(SRC.isLocal = TRUE)  then
    DBMS_OUTPUT.put_line('local is set true');
 else
    DBMS_OUTPUT.put_line('local is set false');
 end if;
END;
/
```

6.2.3 ORDSource Methods Associated with the updateTime Attribute

This section presents reference information on the ORDSource methods associated with the updateTime attribute.

getUpdateTime Method

Format

```
getUpdateTime RETURN DATE;
```

Description

Returns the value of the updateTime attribute for the ORDSource object. This is the timestamp when the object was last changed, or what the user explicitly set by calling the setUpdateTime( ) method.

Parameters

none

Usage

none

Pragmas

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

Exception

none

Example

Get the current value of the updateTime attribute for some data:

```
DECLARE
 SRC ORDSYS.ORDSource;
BEGIN
 SELECT S INTO SRC FROM TS WHERE N = 1 FOR UPDATE;
 SRC.setUpdateTime(SYSDATE);
 UPDATE TS SET S=SRC WHERE N = 1;
 COMMIT;
 SELECT S INTO SRC FROM TS WHERE N = 1 ;
 DBMS_OUTPUT.PUT_LINE(TO_CHAR(SRC.getUpdateTime,'MM-DD-YYYY HH24:MI:SS'));
END;
/
```

setUpdateTime( ) Method

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

If current_time is NULL, updateTime is set to SYSDATE (the current time).

Pragmas

none

Exception

none

Example

See the example in getUpdateTime Method on page 6-14

6.2.4 ORDSource Methods Associated with the srcType, srcLocation, and srcName Attributes

This section presents reference information on the ORDSource methods associated with the srcType, srcLocation, and srcName attributes.

setSourceInformation( ) Method

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 that describes the external data source.

Parameters

source_type
The source type of the external data. See the "ORDSource Object Type" definition in this chapter for more information.

source_location
The source location of the external data. See the "ORDSource Object Type" definition in this chapter for more information.

source_name
The source name of the external data. See the "ORDSource Object Type" definition in this chapter for more information.

Usage

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 exists or is created before you use this method.

Pragmas

none

Exception

If the value for source_type is NULL, then this method raises an INCOMPLETE_SOURCE_INFORMATION exception.

Example

Set the source to point to a file:

```
DECLARE
 SRC ORDSYS.ORDSource;
BEGIN
 SELECT S INTO SRC FROM TS WHERE N = 1 FOR UPDATE;
 SRC.setSourceInformation('FILE','AUDIODIR','testaud.dat');
 DBMS_OUTPUT.PUT_LINE(SRC.getSourceInformation);
 DBMS_OUTPUT.PUT_LINE(SRC.getSourceType);
 DBMS_OUTPUT.PUT_LINE(SRC.getSourceLocation);
 DBMS_OUTPUT.PUT_LINE(SRC.getSourceName);
 UPDATE TS SET S=SRC WHERE N = 1;
 COMMIT;
END;
/
```

getSourceInformation Method

Format

```
getSourceInformation RETURN VARCHAR2;
```

Description

Returns a URL formatted string containing complete information about the external data source.

Parameters

none

Usage

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

Example

See the example in setSourceInformation( ) Method on page 6-18.

getSourceType Method

Format

```
getSourceType RETURN VARCHAR2;
```

Description

Returns the external data source type.

Parameters

none

Usage

This method returns the current value of the srcType attribute, for example 'FILE'.

Pragmas

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

Exception

none

Example

See the example in setSourceInformation( ) Method on page 6-18.

getSourceLocation Method

Format

```
getSourceLocation RETURN VARCHAR2;
```

Description

Returns the external data source location.

Parameters

none

Usage

This method returns the current value of the srcLocation attribute, for example 'BFILEDIR'.

Pragmas

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

Exception

If the value of srcLocation is NULL, then calling this method raises an INCOMPLETE_SOURCE_LOCATION exception.

Example

See the example in setSourceInformation( ) Method on page 6-18.

getSourceName Method

Format

```
getSourceName RETURN VARCHAR2;
```

Description

Returns the external data source name.

Parameters

none

Usage

This method returns the current value of the srcName attribute, for example 'testaud.dat'.

Pragmas

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

Exception

If the value of srcName is NULL, then calling this method raises an INCOMPLETE_SOURCE_NAME exception.

Example

See the example in setSourceInformation( ) Method on page 6-18.

getBFile Method

Format

```
getBFile RETURN BFILE;
```

Description

Returns a BFILE handle, if the srcType is 'FILE'.

Parameters

none

Usage

This method can only be used for a srcType of 'FILE' or BFILE sources.

Pragmas

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

Exception

If the value of srcType is NULL, then calling this method raises an INCOMPLETE_SOURCE_INFORMATION exception.

If the value of srcType is other than FILE, then calling this method raises an INVALID_SOURCE_TYPE exception.

Example

Get a BFILE:

```
DECLARE
    SRC ORDSYS.ORDSource;
    file_handle BFILE;
BEGIN
    SELECT S INTO SRC FROM TS WHERE N = 1 ;
    src.setSourceInformation('FILE','BFILEDIR','testaud.dat');
    file_handle  := SRC.getBFile;
    DBMS_OUTPUT.put_line(DBMS_LOB.GETLENGTH(file_handle));
END;
/
```

6.2.5 ORDSource Methods Associated with Import and Export Operations

This section presents reference information on the ORDSource methods associated with import and export operations.

import( ) Method

Format

```
import(
       ctx      IN OUT RAW,
       dlob     IN OUT NOCOPY BLOB,
       mimetype OUT VARCHAR2,
       format   OUT VARCHAR2);
```

Description

Transfers data from an external data source (specified by first calling setSourceInformation( )) to a local source within an Oracle database.

Parameters

ctx
The source plug-in context information.This information is passed along uninterpreted to the source plug-in handling the import( ) call.

dlob
The destination large object or data object.

mimetype
Out parameter to receive the MIME type of the data, if any, for example, 'audio/basic'.

format
Out parameter to receive the format of the data, if any, for example, 'AUFF'.

Usage

Call setSourceInformation( ) to set the srcType, srcLocation, and srcName attribute information to describe where the data source is located prior to calling the import( ) method.

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

Calling this method uses the ORDPLUGINS.ORDX_<srcType>_SOURCE plug-in package.

Pragmas

none

Exception

If the value of srcType is NULL, then calling this method raises an INCOMPLETE_SOURCE_INFORMATION exception.

If the value of dlob is NULL, then calling this method raises a NULL_SOURCE exception.

If the import( ) method is not supported by the source plug-in being used, a METHOD_NOT_SUPPORTED exception is raised.

Calling this method within a source plug-in when any other exception is raised, raises a SOURCE_PLUGIN_EXCEPTION exception.

Example

Import data from an external data source into the local source and check for exceptions:

```
DECLARE
  SRC ORDSYS.ORDSource;
  mType VARCHAR2(4000);
  format VARCHAR2(4000);
  dblob BLOB;
  ctx RAW(4000) :=NULL;
BEGIN
  SELECT S INTO SRC FROM TS WHERE N = 1 FOR UPDATE;
  SRC.setSourceInformation('FILE','BFILEDIR','testaud.dat');
  SRC.import(ctx,dblob,mType,format);
  COMMIT;
EXCEPTION
   WHEN ORDSYS.ORDSourceExceptions.METHOD_NOT_SUPPORTED THEN
    DBMS_OUTPUT.put_line('Source METHOD_NOT_SUPPORTED caught');
   WHEN ORDSYS.ORDSourceExceptions.SOURCE_PLUGIN_EXCEPTION THEN
    DBMS_OUTPUT.put_line('SOURCE PLUGIN EXCEPTION caught');
   WHEN OTHERS THEN
    DBMS_OUTPUT.PUT_LINE('OTHER EXCEPTION caught');
END;
/
```

importFrom( ) Method

Format

```
importFrom(
          ctx             IN OUT RAW,
          dlob            IN OUT NOCOPY BLOB,
          mimetype        OUT VARCHAR2,
          format          OUT VARCHAR2
          source_type     IN VARCHAR2,
          source_location IN VARCHAR2,
          source_name     IN VARCHAR2);
```

Description

Transfers data from the specified external data source (type, location, name) to a local source within an Oracle 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.

dlob
The destination large object or data object.

mimetype
Out parameter to receive the MIME type of the data, if any, for example, 'audio/basic'.

format
Out parameter to receive the format of the data, if any, for example, 'AUFF'.

source_type
Source type from where the data is to be imported. This also sets the srcType attribute.

source_location
Source location from where the data is to be imported. This also sets the srcLocation attribute.

source_name
Name of the source to be imported. This also sets the srcName attribute.

Usage

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.

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

This method is a combination of a setSourceInformation( ) call followed by an import( ) call.

Calling this method uses the ORDPLUGINS.ORDX_<srcType>_SOURCE plug-in package.

Pragmas

none

Exception

If the value of dlob is NULL, then calling this method raises a NULL_SOURCE exception.

If the importFrom( ) method is not supported by the source plug-in being used, a METHOD_NOT_SUPPORTED exception is raised.

Calling this method within a source plug-in when any other exception is raised, raises a SOURCE_PLUGIN_EXCEPTION exception.

Example

Import data from the specified data source into a BLOB "l":

```
DECLARE
  SRC ORDSYS.ORDSource;
  mType  VARCHAR2(4000);
  format VARCHAR2(4000);
  l   BLOB;
  ctx RAW(4000) :=NULL;
BEGIN
  SELECT S INTO SRC FROM TS WHERE N = 1 FOR UPDATE;
  SRC.importFrom(ctx, l, mType, format,'FILE','AUDIODIR','testaud.dat');
  DBMS_OUTPUT.PUT_LINE(TO_CHAR(SRC.getContentLength(ctx)));
  UPDATE TS SET S=SRC WHERE N=1;
```

```
  COMMIT;
END;
```

export( ) Method

Format

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

Description

Transfers data from a local source (localData) within an Oracle database to an external data source.

> **Note:** This method is supported only for external sources.

Parameters

ctx
The source plug-in context information.

source_type
The source type of the location to where data is to be exported.

source_location
The location where the data is to be exported.

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

Usage

This method exports data out of the localData to another source.

Calling this method uses the ORDPLUGINS.ORDX_<srcType>_SOURCE plug-in package.

After exporting data, the srcType, srcLocation, and srcName attributes are updated with input parameter values. After calling the export method, call the deleteLocalContent method to delete the content of the local data if you want to delete the contents of the localData attribute.

Pragmas

none

Exception

If the value of srcType is NULL, then calling this method raises an INCOMPLETE_SOURCE_INFORMATION exception.

If the export( ) method is not supported by the source plug-in being used, a METHOD_NOT_SUPPORTED exception is raised.

Calling this method within a source plug-in when any other exception is raised, raises a SOURCE_PLUGIN_EXCEPTION exception.

Example

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

```
DECLARE
  obj ORDSYS.ORDSource;
  ctx RAW(4000) :=NULL;
BEGIN
   SELECT S INTO obj FROM TS   WHERE N = 1;
   obj.export(ctx,'FILE','AUDIODIR','testaud.dat');
  EXCEPTION
  WHEN ORDSYS.ORDSourceExceptions.METHOD_NOT_SUPPORTED THEN
   DBMS_OUTPUT.put_line('Source METHOD_NOT_SUPPORTED caught');
  WHEN ORDSYS.ORDSourceExceptions.SOURCE_PLUGIN_EXCEPTION THEN
   DBMS_OUTPUT.put_line('SOURCE PLUGIN EXCEPTION caught');
  WHEN OTHERS THEN
   DBMS_OUTPUT.put_line('OTHER EXCEPTION caught');
END;
/
```

6.2.6 ORDSource Methods Associated with the localData Attribute

This section presents reference information on the ORDSource methods associated with the localData attribute.

getContentLength( ) Method

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 a local BLOB data source, the length is returned as a number of bytes. The unit type of the returned value is defined by the plug-in that implements this method.

Parameters

ctx
The source plug-in context information.

Usage

This method is not supported for all source types. For example, HTTP type sources do not support this method. If you want to implement this call for HTTP type sources, you must define your own modified HTTP source plug-in and implement this method.

Calling this method uses the ORDPLUGINS.ORDX_<srcType>_SOURCE plug-in package.

Pragmas

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

Exception

If the value of srcType is NULL and data is not stored locally in the BLOB, then calling this method raises an INCOMPLETE_SOURCE_INFORMATION exception.

Calling this method within a source plug-in when any other exception is raised, raises a SOURCE_PLUGIN_EXCEPTION exception.

Example

Get the length of the data content stored in the source:

```
DECLARE
  SRC ORDSYS.ORDSource;
  ctx RAW(4000) :=NULL;
BEGIN
  SELECT S INTO SRC FROM TS WHERE N = 1;
  DBMS_OUTPUT.PUT_LINE(SRC.getSourceInformation);
  DBMS_OUTPUT.PUT_LINE(TO_CHAR(SRC.getContentLength(ctx)));
  EXCEPTION
   WHEN ORDSYS.ORDSourceExceptions.INCOMPLETE_SOURCE_INFORMATION THEN
    DBMS_OUTPUT.PUT_LINE('Source not specified');
 END;
/
```

getSourceAddress( ) Method

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 only implemented for user-defined sources.

Parameters

ctx
The source plug-in context information.

userData
Information input by the user needed by some sources to obtain the desired source address.

Usage

Use this method to return the address of an external data source when the source needs to format this information in some unique way. For example, call the get-SourceAddress( )method to obtain the address for Real Networks server sources or URLs containing data sources located on Oracle Application Server.

Calling this method uses the ORDPLUGINS.ORDX_<srcType>_SOURCE plug-in package.

Pragmas

Pragma RESTRICT_REFERENCES(getSourceAddress, WNDS, WNPS, RNDS, RNPS)

Exception

If the value of srcType is NULL, then calling this method raises an INCOMPLETE_SOURCE_INFORMATION exception.

Calling this method within a source plug-in when any other exception is raised, raises a SOURCE_PLUGIN_EXCEPTION exception.

Example

Get the source address for the external source:

```
DECLARE
  SRC ORDSYS.ORDSource;
  ctx RAW(4000) :=NULL;
  userData VARCHAR2(4000);
BEGIN
  SELECT S INTO SRC FROM TS WHERE N = 1 FOR UPDATE;
  -- process tempBlob
  SRC.setSourceInformation('FILE','AUDIODIR','testaud.dat');
  userData :=NULL;
  DBMS_OUTPUT.PUT_LINE(SRC.getSourceAddress(ctx,userData));
  COMMIT;
END;
/
```

getLocalContent Method

Format

```
getLocalContent RETURN BLOB;
```

Description

Returns the content or BLOB handle of the local data.

Parameters

none

Usage

none

Pragmas

Pragma RESTRICT_REFERENCES(getLocalContent, WNDS, WNPS, RNDS, RNPS)

Exception

none

Example

Get the local content for a local source prior to an import operation:

```
DECLARE
 SRC ORDSYS.ORDSource;
 l   BLOB;
 mimeType VARCHAR2(4000);
 format VARCHAR2(4000);
 ctx RAW(4000) := NULL;
BEGIN
 SELECT S INTO SRC FROM TS WHERE N = 1 FOR UPDATE;
 l := SRC.getLocalContent;
 SRC.importFrom(ctx, 1, mimeType, format,'FILE','AUDIODIR','testaud.dat');
 DBMS_OUTPUT.PUT_LINE(TO_CHAR(DBMS_LOB.GETLENGTH(1)));
 UPDATE TS SET S=SRC WHERE N=1;
 COMMIT;
END;
```

getContentInTempLob( ) Method

Format

```
getContentInTempLob(
                    ctx      IN OUT RAW,
                    tempLob  IN OUT NOCOPY BLOB,
                    mimetype OUT VARCHAR2,
                    format   OUT VARCHAR2,
                    duration IN PLS_INTEGER := 10,
                    cache    IN BOOLEAN := TRUE);
```

Description

Transfers data from the current data source into a temporary LOB, which will be allocated and initialized as a part of this call.

Parameters

ctx
The source plug-in context information.

tempLob
Uninitialized BLOB locator, which will be allocated in this call.

mimetype
Out parameter to receive the MIME type of the data, for example, 'audio/basic'.

format
Out parameter to receive the format of the data, for example, 'AUFF'.

duration
The life of the temporary LOB to be allocated. The life of the temporary LOB can be for the duration of the call, the transaction, or for the session. The default is DBMS_LOB.SESSION. Valid values for each duration state are as follows:

DBMS_LOB.CALL

DBMS_LOB.TRANSACTION

DBMS_LOB.SESSION

cache
Whether or not you want to keep the data cached. The value is either TRUE or FALSE. The default is TRUE.

Usage

none

Pragmas

none

Exception

When working with temporary LOBs, a NO_DATA_FOUND exception can be raised for looping read operations that reach the end of the LOB, and there are no more bytes to be read from the LOB.

Calling this method within a source plug-in when any other exception is raised, raises a SOURCE_PLUGIN_EXCEPTION exception.

Example

Get data from an external data source into a temporary LOB on the local source:

```
DECLARE
 SRC ORDSYS.ORDSource;
 userData VARCHAR2(4000);
 l   BLOB;
 tempBLob BLOB;
 mimeType VARCHAR2(4000);
 format VARCHAR2(4000);
 ctx RAW(4000) := NULL;
BEGIN
 SELECT S INTO SRC FROM TS WHERE N = 1 FOR UPDATE;
 SRC.importFrom(ctx, src.localData, mimeType, format,'FILE','AUDIODIR','testaud.dat');
 SRC.getContentInTempLob(ctx,tempBlob,
                                        mimeType,format, 0,FALSE);
-- process tempBlob

 DBMS_OUTPUT.PUT_LINE(TO_CHAR(SRC.getContentLength(ctx)));
 DBMS_OUTPUT.PUT_LINE(SRC.getSourceAddress(ctx,userData));
 DBMS_OUTPUT.PUT_LINE(TO_CHAR(DBMS_LOB.GETLENGTH(tempBlob)));
 UPDATE TS SET S=SRC WHERE N=1;
 COMMIT;
END;
/
```

deleteLocalContent Method

Format

```
deleteLocalContent;
```

Description

Deletes the local data from the current local source (localData).

Parameters

none

Usage

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

Exception

none

Example

Delete the local data from the current local source:

```
DECLARE
 SRC ORDSYS.ORDSource;
BEGIN
 SELECT S INTO SRC FROM TS WHERE N = 1 FOR UPDATE;
 SRC.deleteLocalContent;
 UPDATE TS SET S=SRC WHERE N=1;
 COMMIT;
END;
/
```

6.2.7 ORDSource Methods Associated with File Operations

This section presents reference information on the ORDSource methods associated with accessing an external data source.

open( ) Method

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 argument.

ctx
The source plug-in context information.

Usage

Calling this method uses the ORDPLUGINS.ORDX_<srcType>_SOURCE plug-in package.

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

Pragmas

none

Exception

If the value for srcType is NULL and data is not local, then calling this method raises an INCOMPLETE_SOURCE_INFORMATION exception.

If the open( ) method is not supported by the source plug-in being used, a METHOD_NOT_SUPPORTED exception is raised.

Calling this method within a source plug-in when any other exception is raised, raises a SOURCE_PLUGIN_EXCEPTION exception.

Example

Open an external data source:

```
DECLARE
  SRC ORDSYS.ORDSource;
  res INTEGER;
  ctx RAW(4000) :=NULL;
  userArg RAW(4000);
BEGIN
  SELECT S INTO SRC FROM TS WHERE N = 1 FOR UPDATE;
  res := SRC.open(userArg, ctx);
  -- manipulate the source
  res := SRC.close(ctx);
  EXCEPTION
   WHEN ORDSYS.ORDSourceExceptions.METHOD_NOT_SUPPORTED THEN
    DBMS_OUTPUT.PUT_LINE('Source not specified');
   WHEN OTHERS THEN
    DBMS_OUTPUT.PUT_LINE('Exception caught');
END;
/
```

close( ) Method

Format

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

Description

Closes a data source.

Parameters

ctx
The source plug-in context information.

Usage

Calling this method uses the ORDPLUGINS.ORDX_<srcType>_SOURCE plug-in package.

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

Pragmas

none

Exception

If the value for srcType is NULL and data is not local, then calling this method raises an INCOMPLETE_SOURCE_INFORMATION exception.

If the close( ) method is not supported by the source plug-in being used, a METHOD_NOT_SUPPORTED exception is raised.

Calling this method within a source plug-in when any other exception is raised, raises a SOURCE_PLUGIN_EXCEPTION exception.

Example

Close an external data source:

```
DECLARE
  SRC ORDSYS.ORDSource;
  res INTEGER;
  ctx RAW(4000) :=NULL;
BEGIN
  SELECT S INTO SRC FROM TS WHERE N = 1 FOR UPDATE;
  res := SRC.close(ctx);
  UPDATE TS SET S=SRC WHERE N=1;
  COMMIT;
  EXCEPTION
   WHEN ORDSYS.ORDSourceExceptions.METHOD_NOT_SUPPORTED THEN
    DBMS_OUTPUT.PUT_LINE('Source not specified');
   WHEN OTHERS THEN
    DBMS_OUTPUT.PUT_LINE('Exception caught');
END;
/
```

trim( ) Method

Format

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

Description

Trims a data source.

Parameters

ctx
The source plug-in context information.

newlen
The trimmed new length.

Usage

Calling this method uses the ORDPLUGINS.ORDX_<srcType>_SOURCE plug-in package.

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

Pragmas

none

Exception

If the value for srcType is NULL and data is not local, then calling this method raises an INCOMPLETE_SOURCE_INFORMATION exception.

If the trim( ) method is not supported by the source plug-in being used, a METHOD_NOT_SUPPORTED exception is raised.

Calling this method within a source plug-in when any other exception is raised, raises a SOURCE_PLUGIN_EXCEPTION exception.

Example

Trim an external data source:

```
DECLARE
  SRC ORDSYS.ORDSource;
  res INTEGER;
  ctx RAW(4000) :=NULL;
BEGIN
  SELECT S INTO SRC FROM TS WHERE N = 1 FOR UPDATE;
  res := SRC.trim(ctx,0);
  UPDATE TS SET S=SRC WHERE N=1;
  COMMIT;
  EXCEPTION
   WHEN ORDSYS.ORDSourceExceptions.METHOD_NOT_SUPPORTED THEN
    DBMS_OUTPUT.PUT_LINE('Source not specified');
   WHEN OTHERS THEN
    DBMS_OUTPUT.PUT_LINE('Exception caught');
END;
/
```

6.2.8 ORDSource Methods Associated with Read/Write Operations

This section presents reference information on the ORDSource methods associated with read / write operations.

read( ) Method

Format

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

Description

Allows you to read a buffer of numBytes from a source beginning at a start position (startPos).

Parameters

ctx
The source plug-in context information.

startPos
The start position in the data source.

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

buffer
The buffer to where the data will be read.

Usage

This method is not supported for HTTP sources.

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

Calling this method uses the ORDPLUGINS.ORDX_<srcType>_SOURCE plug-in package.

Pragmas

none

Exception

If data is stored locally and localData is NULL, then calling this method raises a NULL_SOURCE exception.

If the value of srcType is NULL and data is not local, then calling this method raises an INCOMPLETE_SOURCE_INFORMATION exception.

If the read( ) method is not supported by the source plug-in being used, a METHOD_NOT_SUPPORTED exception is raised.

Calling this method within a source plug-in when any other exception is raised, raises a SOURCE_PLUGIN_EXCEPTION exception.

Example

Read a buffer from the source:

```
DECLARE
  SRC ORDSYS.ORDSource;
  i INTEGER;
  buffer RAW(4000);
  ctx RAW(4000) :=NULL;
BEGIN
  i := 20;
  SELECT S INTO SRC FROM TS WHERE N = 1 ;
  SRC.read(ctx, 1, i, buffer);
END;
/
```

write( ) Method

Format

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

Description

Allows you to write a buffer of numBytes to a source beginning at a start position (startPos).

Parameters

ctx
The source plug-in context information.

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

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

buffer
The buffer of data to be written.

Usage

This method assumes that the writable source allows you to write numBytes at a random byte location. For example, the FILE and HTTP source types are not writable sources and do not support this method.

Calling this method uses the ORDPLUGINS.ORDX_<srcType>_SOURCE plug-in package.

Pragmas

none

Exception

If local is 1 or NULL and localData is NULL, then calling this method raises a NULL_SOURCE exception.

If the value of srcType is NULL and data is not local, then calling this method raises an INCOMPLETE_SOURCE_INFORMATION exception.

If the write( ) method is not supported by the source plug-in being used, a METHOD_NOT_SUPPORTED exception is raised.

Calling this method within a source plug-in when any other exception is raised, raises a SOURCE_PLUGIN_EXCEPTION exception.

Example

Write a buffer to the source:

```
DECLARE
  SRC ORDSYS.ORDSource;
  n INTEGER := 6;
  ctx RAW(4000) :=NULL;
BEGIN
  SELECT S INTO SRC FROM TS WHERE N = 1 FOR UPDATE;
  DBMS_OUTPUT.PUT_LINE(SRC.getSourceInformation);
  SRC.write(ctx, 1, n, UTL_RAW.CAST_TO_RAW('helloP'));
  DBMS_OUTPUT.PUT_LINE(TO_CHAR(SRC.getContentLength(ctx)));
  COMMIT;
END;
/
```

6.2.9 ORDSource Methods Associated with Processing Commands to the External Source

This section presents reference information on the ORDSource methods associated with processing commands to the external source.

processCommand( ) Method

Format

```
processCommand(
               ctx     IN OUT RAW,
               command IN VARCHAR2,
               arglist IN VARCHAR2,
               result  OUT RAW)
RETURN RAW;
```

Description

Allows you to send commands and related arguments to the source plug-in. This method is supported only for user-defined sources.

Parameters

ctx
The source plug-in context information.

command
Any command recognized by the source plug-in.

arglist
The arguments for the command.

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

Usage

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

Exception

If the value of srcType is NULL, then calling this method raises an INCOMPLETE_SOURCE_INFORMATION exception.

If the processCommand( ) method is not supported by the source plug-in being used, a METHOD_NOT_SUPPORTED exception is raised.

Calling this method within a source plug-in when any other exception is raised, raises a SOURCE_PLUGIN_EXCEPTION exception.

Example

Process some commands:

```
DECLARE
  obj ORDSYS.ORDSource ;
  res RAW(4000);
  result RAW(4000);
  command VARCHAR2(4000);
  argList VARCHAR2(4000);
  ctx RAW(4000) :=NULL;
BEGIN
 select s into obj from TS where N =1 for UPDATE;
  command := 'xxx ';
  argList := 'yyy ';
  res := obj.processCommand(ctx, command, argList, result);
  UPDATE TS SET s=obj WHERE N=1 ;
  COMMIT;
  EXCEPTION
   WHEN ORDSYS.ORDSourceExceptions.METHOD_NOT_SUPPORTED THEN
    DBMS_OUTPUT.put_line('Source METHOD_NOT_SUPPORTED caught');
   WHEN ORDSYS.ORDSourceExceptions.SOURCE_PLUGIN_EXCEPTION THEN
    DBMS_OUTPUT.put_line('SOURCE PLUGIN EXCEPTION caught');
   WHEN OTHERS THEN
    DBMS_OUTPUT.put_line('OTHER EXCEPTION caught');
END;
/
```

6.3 Packages or PL/SQL Plug-ins

This section presents reference information on the packages or PL/SQL plug-ins provided.

Any method invoked from a source plug-in call has the first argument as obj (ORDSource) and the second argument as ctx (RAW).

Plug-ins must be named as ORDX_<name>_<module_name> where the <module_name> is SOURCE for ORDSource. For example, the FILE plug-in described in Section 6.3.1, is named ORDX_FILE_SOURCE and <name> is the source type.

Exceptions must be raised from and recorded in a package named as ORD_<module_name>Exceptions. For example, ORDSource exceptions are raised and recorded in a package named ORDSourceExceptions (see Appendix H).

6.3.1 ORDPLUGINS.ORDX_FILE_SOURCE Package

The ORDPLUGINS.ORDX_FILE_SOURCE package or PL/SQL plug-in is provided.

```
CREATE OR REPLACE PACKAGE ORDX_FILE_SOURCE AS
  -- functions/procedures
  FUNCTION processCommand(obj     IN OUT NOCOPY ORDSYS.ORDSource,
                          ctx     IN OUT RAW,
                          cmd     IN VARCHAR2,
                          arglist IN VARCHAR2,
                          result  OUT RAW)
           RETURN RAW;
  PROCEDURE import(obj      IN OUT NOCOPY ORDSYS.ORDSource,
                   ctx      IN OUT RAW,
                   dlob     IN OUT NOCOPY BLOB,
                   mimetype OUT VARCHAR2,
                   format   OUT VARCHAR2);
  PROCEDURE importFrom(obj      IN OUT NOCOPY ORDSYS.ORDSource,
                       ctx      IN OUT RAW,
                       dlob     IN OUT NOCOPY BLOB,
                       mimetype OUT VARCHAR2,
                       format   OUT VARCHAR2,
                       loc      IN VARCHAR2,
                       name     IN VARCHAR2);
  PROCEDURE export(obj  IN OUT NOCOPY ORDSYS.ORDSource,
                   ctx  IN OUT RAW,
                   dlob IN OUT NOCOPY BLOB,
                   loc  IN VARCHAR2,
                   name IN VARCHAR2);
```

```
  FUNCTION  getContentLength(obj  IN ORDSYS.ORDSource,
                             ctx  IN OUT RAW),
            RETURN INTEGER;
  PRAGMA RESTRICT_REFERENCES(getContentLength, WNDS, WNPS, RNDS, RNPS);
  FUNCTION  getSourceAddress(obj  IN ORDSYS.ORDSource,
                             ctx  IN OUT RAW,
                             userData IN VARCHAR2)
            RETURN VARCHAR2;
  PRAGMA RESTRICT_REFERENCES(getSourceAddress, WNDS, WNPS, RNDS, RNPS);
  FUNCTION open(obj IN OUT NOCOPY ORDSYS.ORDSource, userArg IN RAW,
         ctx OUT RAW) RETURN INTEGER;
  FUNCTION close(obj IN OUT NOCOPY ORDSYS.ORDSource, ctx IN OUT RAW)
         RETURN INTEGER;
  FUNCTION trim(obj IN OUT NOCOPY ORDSYS.ORDSource,
         ctx IN OUT RAW,
         newlen IN INTEGER) RETURN INTEGER;
PROCEDURE read(obj     IN OUT NOCOPY ORDSYS.ORDSource,
               ctx      IN OUT RAW,
               startPos IN INTEGER,
               numBytes IN OUT INTEGER,
               buffer   OUT RAW);
PROCEDURE write(obj      IN OUT NOCOPY ORDSYS.ORDSource,
                ctx      IN OUT RAW,
                startPos IN INTEGER,
                numBytes IN OUT INTEGER,
                buffer   OUT RAW);
END ORDX_FILE_SOURCE;
/
```

Table 6–1 shows the methods supported in the ORDX_FILE_SOURCE package and the exceptions raised if you call a method that is not supported.

Table 6–1 Methods Supported in the ORDPLUGINS.ORDX_FILE_SOURCE Package

Name of Method	Level of Support
processCommand	Not supported - raises exception: METHOD_NOT_SUPPORTED
import	Supported
importFrom	Supported
export	Not supported - raises exception: METHOD_NOT_SUPPORTED
getContentLength	Supported
getSourceAddress	Supported

Table 6–1 Methods Supported in the ORDPLUGINS.ORDX_FILE_SOURCE Package (Cont.)

Name of Method	Level of Support
open	Supported
close	Supported
trim	Not supported - raises exception: METHOD_NOT_SUPPORTED
read	Supported
write	Not supported - raises exception: METHOD_NOT_SUPPORTED

6.3.2 ORDPLUGINS.ORDX_HTTP_SOURCE Package

The ORDPLUGINS.ORDX_HTTP_SOURCE package or PL/SQL plug-in is provided.

```
CREATE OR REPLACE PACKAGE ORDX_HTTP_SOURCE AS
  -- functions/procedures
  FUNCTION processCommand(obj     IN OUT NOCOPY ORDSYS.ORDSource,
                          ctx     IN OUT RAW,
                          cmd     IN VARCHAR2,
                          arglist IN VARCHAR2,
                          result  OUT RAW)
           RETURN RAW;
  PROCEDURE import(obj      IN OUT NOCOPY ORDSYS.ORDSource,
                   ctx      IN OUT RAW,
                   dlob     IN OUT NOCOPY BLOB,
                   mimetype OUT VARCHAR2,
                   format   OUT VARCHAR2);
  PROCEDURE importFrom(obj      IN OUT NOCOPY ORDSYS.ORDSource,
                       ctx      IN OUT RAW,
                       dlob     IN OUT NOCOPY BLOB,
                       mimetype OUT VARCHAR2,
                       format   OUT VARCHAR2,
                       loc      IN VARCHAR2,
                       name     IN VARCHAR2);
  PROCEDURE export(obj  IN OUT NOCOPY ORDSYS.ORDSource,
                   ctx  IN OUT RAW,
                   dlob IN OUT NOCOPY BLOB,
                   loc  IN VARCHAR2,
                   name IN VARCHAR2);
  FUNCTION  getContentLength(obj  IN ORDSYS.ORDSource,
                             ctx  IN OUT RAW)
            RETURN INTEGER;
  PRAGMA RESTRICT_REFERENCES(getContentLength, WNDS, WNPS, RNDS, RNPS);
```

```
  FUNCTION  getSourceAddress(obj  IN ORDSYS.ORDSource,
                             ctx  IN OUT RAW,
                             userData IN VARCHAR2)
            RETURN VARCHAR2;
  PRAGMA RESTRICT_REFERENCES(getSourceAddress, WNDS, WNPS, RNDS, RNPS);
  FUNCTION open(obj IN OUT NOCOPY ORDSYS.ORDSource, userArg IN RAW,
           ctx OUT RAW) RETURN INTEGER;
  FUNCTION close(obj IN OUT NOCOPY ORDSYS.ORDSource, ctx IN OUT RAW)
           RETURN INTEGER;
  FUNCTION trim(obj IN OUT NOCOPY ORDSYS.ORDSource,
           ctx IN OUT RAW,
           newlen IN INTEGER) RETURN INTEGER;
  PROCEDURE read(obj      IN OUT NOCOPY ORDSYS.ORDSource,
                 ctx      IN OUT RAW,
                 startPos IN INTEGER,
                 numBytes IN OUT INTEGER,
                 buffer   OUT RAW);
  PROCEDURE write(obj      IN OUT NOCOPY ORDSYS.ORDSource,
                  ctx      IN OUT RAW,
                  startPos IN INTEGER,
                  numBytes IN OUT INTEGER,
                  buffer   OUT RAW);
END ORDX_HTTP_SOURCE;
/
```

Table 6–2 shows the methods supported in the ORDX_HTTP_SOURCE package and the exceptions raised if you call a method that is not supported.

Table 6–2 Methods Supported in the ORDPLUGINS.ORDX_HTTP_SOURCE Package

Name of Method	Level of Support
processCommand	Not supported - raises exception: METHOD_NOT_SUPPORTED
import	Supported
importFrom	Supported
export	Not supported - raises exception: METHOD_NOT_SUPPORTED
getContentLength	Not supported - raises exception: METHOD_NOT_SUPPORTED
getSourceAddress	Supported
open	Supported
close	Supported

Table 6–2 Methods Supported in the ORDPLUGINS.ORDX_HTTP_SOURCE Package (Cont.)

Name of Method	Level of Support
trim	Not supported - raises exception: METHOD_NOT_SUPPORTED
read	Not supported - raises exception: METHOD_NOT_SUPPORTED
write	Not supported - raises exception: METHOD_NOT_SUPPORTED

6.3.3 ORDPLUGINS.ORDX_<srcType>_SOURCE Package

Use the ORDPLUGINS.ORDX_<srcType>_SOURCE package or PL/SQL plug-in as a template to create your own source type. Use the ORDPLUGINS.ORDX_FILE_SOURCE and ORDPLUGINS.ORDX_HTTP_SOURCE packages as a guide in developing your new source type package.

6.3.4 Extending *inter*Media to Support a New Data Source

Extending *inter*Media to support a new data source consists of four steps:

1. Design your new data source.
2. Implement your new data source and name it, for example, ORDX_MY_SOURCE.SQL.
3. Install your new ORDX_MY_SOURCE.SQL plug-in in the ORDPLUGINS schema.
4. Grant EXECUTE privileges on your new plug-in, for example, ORDX_MY_SOURCE.SQL plug-in to PUBLIC.

Section 2.4 briefly describes how to extend *inter*Media to support a new data source for audio and video data and describe the interfaces. A package body listing is provided in Example 6–1 to assist you in this operation. Add your variables to the places that say "--Your variables go here" and add your code to the places that say "--Your code goes here".

Example 6–1 Package Body Listing for Extending Support to a New Data Source

```
CREATE OR REPLACE PACKAGE BODY ORDX_MY_SOURCE
AS
  -- functions/procedures
  FUNCTION processCommand(
                    obj  IN OUT NOCOPY ORDSYS.ORDSource,
                    ctx  IN OUT RAW,
                    cmd  IN VARCHAR2,
                    arglist IN VARCHAR2,
```

```
                result OUT RAW)
RETURN RAW
IS
--Your variables go here
BEGIN
--Your code goes here
END processCommand;
PROCEDURE import( obj  IN OUT NOCOPY ORDSYS.ORDSource,
                  ctx  IN OUT RAW,
                  dlob IN OUT NOCOPY BLOB,
                  mimetype OUT VARCHAR2,
                  format   OUT VARCHAR2)
IS
--Your variables go here
BEGIN
--Your code goes here
END import;
PROCEDURE importFrom( obj      IN OUT NOCOPY ORDSYS.ORDSource,
                      ctx      IN OUT RAW,
                      dlob     IN OUT NOCOPY BLOB,
                      mimetype OUT VARCHAR2,
                      format   OUT VARCHAR2,
                      loc      IN VARCHAR2,
                      name     IN VARCHAR2)
IS
--Your variables go here
BEGIN
--Your code goes here
END importFrom;
PROCEDURE export( obj  IN OUT NOCOPY ORDSYS.ORDSource,
                  ctx  IN OUT RAW,
                  dlob IN OUT NOCOPY BLOB,
                  loc  IN VARCHAR2,
                  name IN VARCHAR2)
IS
--Your variables go here
BEGIN
--Your code goes here
END export;

FUNCTION  getContentLength( obj  IN ORDSYS.ORDSource,
                            ctx  IN OUT RAW)
RETURN INTEGER
IS
--Your variables go here
```

```
BEGIN
--Your code goes here
END getContentLength;
FUNCTION  getSourceAddress(obj  IN ORDSYS.ORDSource,
                           ctx  IN OUT RAW,
                           userData IN VARCHAR2)
RETURN VARCHAR2
IS
--Your variables go here
BEGIN
--Your code goes here
END getSourceAddress;
FUNCTION open(obj IN OUT NOCOPY ORDSYS.ORDSource, userArg IN RAW, ctx OUT RAW)
RETURN INTEGER
IS
--Your variables go here
BEGIN
--Your code goes here
END open;
FUNCTION close(obj IN OUT NOCOPY ORDSYS.ORDSource, ctx IN OUT RAW)
RETURN INTEGER
IS
--Your variables go here
BEGIN
--Your code goes here
END close;
FUNCTION trim(obj    IN OUT NOCOPY ORDSYS.ORDSource,
                     ctx    IN OUT RAW,
                     newlen IN INTEGER)
RETURN INTEGER
IS
--Your variables go here
BEGIN
--Your code goes here
END trim;
PROCEDURE read(obj      IN OUT NOCOPY ORDSYS.ORDSource,
               ctx      IN OUT RAW,
               startPos IN INTEGER,
               numBytes IN OUT INTEGER,
               buffer   OUT RAW)
IS
--Your variables go here
BEGIN
--Your code goes here
END read;
```

```
  PROCEDURE write(obj      IN OUT NOCOPY ORDSYS.ORDSource,
                  ctx      IN OUT RAW,
                  startPos IN INTEGER,
                  numBytes IN OUT INTEGER,
                  buffer   OUT RAW)
  IS
  --Your variables go here
  BEGIN
  --Your code goes here
  END write;
END ORDX_MY_SOURCE;
/
show errors;
```

7

Tuning Tips for the DBA

This chapter provides tuning tips for the Oracle DBA who wants to achieve more efficient storage of multimedia data in the database when using Oracle8*i* *inter*Media. Multimedia data consists of a variety of media types including character text, images, audio clips, video clips, line drawings, and so forth. All these media types are typically stored in LOBs, in either internal LOBs (stored in an internal database tablespace) or in external LOBs in operating system files outside of the database tablespaces, in BFILEs.

Internal LOBs consist of: CLOBs, NCLOBs, and BLOBs and can be up to 4 gigabytes in size. BFILEs can be as large as the operating system will allow up to a maximum of 4 gigabytes.

Oracle8*i* *inter*Media manages a variety of LOB types. The following general topics will help you to better manage your *inter*Media LOB data:

- Improving *inter*Media LOB data INSERT performance
- Using other bulk loading methods
- User guidelines for best performance practices
- Improving *inter*Media LOB data retrieval and update performance
- Setting the maximum number of open BFILEs
- Using *inter*Media LOBs in partitioned tables

For more information, see information about LOB partitioning, LOB tuning, and LOB buffering in the *Oracle8i Application Developer's Guide - Large Objects (LOBs)*, *Oracle Call Interface Programmer's Guide*, *Oracle8i Concepts*, and *Oracle8i Tuning* manuals.

For information on any restrictions to consider when using LOBs, see *Oracle8i Application Developer's Guide - Large Objects (LOBs).*

For information on guidelines for using the DIRECTORY feature in Oracle8*i* to enable a simple, flexible, nonintrusive, yet secure mechanism for the DBA to manage access to large files in the server file system, see *Oracle8i Application Developer's Guide - Large Objects (LOBs).*

7.1 Improving Multimedia LOB Data INSERT Performance

Issues to Consider in Creating Tables that Will Contain LOBs

The following information provides some strategies to consider when you create tables that will contain LOB columns. These topics are discussed in more detail and with examples in *Oracle8i Application Developer's Guide - Large Objects (LOBs)*. The information that follows is excerpted in whole or in part from Chapter 2 and is briefly presented to give you an overview of the topic. Refer to *Oracle8i Application Developer's Guide - Large Objects (LOBs)* for the most current information.

- Initializing internal LOBs to NULL or EMPTY

  A LOB value set to NULL has no locator. By contrast, an empty LOB stored in a table is a LOB of zero length that has a locator. So, if you SELECT from an empty LOB column or attribute, you get back a locator, which you can use to fill the LOB with data using the OCI or DBMS_LOB routines.

  – Setting a LOB to NULL

    You may want to set the internal LOB value to NULL upon inserting the row in cases where you do not have the LOB data at the time of the INSERT operation or if you want to issue a SELECT statement at some later time, or for both of these reasons.

    However, the drawback to this approach is that you must then issue a SQL UPDATE statement to reset the NULL LOB column -- to EMPTY_BLOB( ) or EMPTY_CLOB( ) or to a known value for internal LOBs, or to a file name for external LOBs. The point is that you cannot call the OCI or the PL/SQL DBMS_LOB functions on a LOB that is NULL. These functions work only with a locator, and if the LOB column is NULL, there is no locator in the row.

  – Setting an internal LOB to EMPTY

    If you do not want to set an internal LOB column to NULL, another option is for you to set the LOB value to EMPTY by using the function EMPTY_BLOB ( ) or EMPTY_CLOB( ) in the INSERT statement.

Even better, use the RETURNING clause (thereby eliminating a round trip that is necessary for the subsequent SELECT statement). Then, immediately call OCI or the PL/SQL DBMS_LOB functions to fill the LOB with data. See *Oracle8i Application Developer's Guide - Large Objects (LOBs)* for an example.

- Stipulating tablespace and storage characteristics for internal LOBs
  - Tablespace

    The best performance for LOBs can often be achieved by specifying storage for LOBs in a tablespace that is different from the one used for the table that contains the LOB column. If many different LOBs are to be accessed frequently, it may also be useful to specify a separate tablespace for each LOB column or attribute in order to reduce device contention. Preallocate the tablespace to the required allocation size to avoid allocation when inserting BLOB data. See the *Oracle8i SQL Reference* manual for examples, specifically the CREATE TABLE statement and the LOB Column example. See Example 7–1.

  - LOB index

    The LOB index is an internal structure that is strongly associated with the LOB storage. This implies that a user may not delete the LOB index and rebuild it. Note that the LOB index cannot be altered. The system determines which tablespace to use for the LOB data and LOB index depending on the user specification in the LOB storage clause:

    * If you do not specify a tablespace for the LOB data, the table's tablespace is used for the LOB data and index.
    * If you specify a tablespace for the LOB data, both the LOB data and index use the tablespace that was specified.

    If in creating tables in Oracle release 8.1, you specify a tablespace for the LOB index for a nonpartitioned table, your specification of the tablespace will be ignored and the LOB index and the LOB data will use the same tablespace. Partitioned LOBs do not include the LOB index syntax.

    Specifying a separate tablespace for the LOB storage segments will decrease the contention on the tablespace of the table.

  - PCTVERSION option

    When a LOB is modified, a new version of the LOB page is made in order to support a consistent read of prior versions of the LOB value.

PCTVERSION is the percent of all used LOB data space that can be occupied by old versions of LOB data pages. As soon as old versions of LOB data pages start to occupy more than the PCTVERSION amount of used LOB space, Oracle tries to reclaim the old versions and reuses them. In other words, PCTVERSION is the percentage of used LOB data blocks that is available for versioning of old LOB data.

One way of approximating PCTVERSION is to set PCTVERSION = (% of LOBs updated at any given point in time) times (% of each LOB updated whenever a LOB is updated) times (% of LOBs being read at any given point in time). Basically, the idea is to allow for a percentage of LOB storage space to be used as old versions of LOB pages so readers are able to get consistent reads of data that has been updated.

Setting PCTVERSION to twice the default allows more free pages to be used for old versions of data pages. Because large queries may require consistent reads of LOBs, it is useful to keep more old versions of LOB pages around. Of course, LOB storage may increase because Oracle will not be reusing free pages aggressively.

The more infrequent and smaller the LOB updates are, the less space that needs to be reserved for old copies of LOB data. If existing LOBs are known to be read-only, you could safely set PCTVERSION to 0% because there would never be any pages needed for old versions of data.

– CACHE or NOCACHE option

  Use the CACHE option on LOBs if the same LOB data is to be accessed frequently. Use the NOCACHE option (the default) if LOB data is to be read only once, or infrequently.

  Use the NOCACHE option when reading and writing large quantities of LOB data to reduce the I/O operations.

  See Example 7–1.

– LOGGING or NOLOGGING option

  An example of when NOLOGGING is useful is with bulk loads or inserts of data. See Example 7–1. For instance, when loading data into the LOB, if you do not care about redo logging and can just start the load over if it fails, set the LOB's data segment storage characteristics to NOCACHE NOLOGGING. This setting gives good performance for the initial load of data. Once you have completed loading the data, you can use the ALTER TABLE statement to modify the LOB storage characteristics for the LOB data segment to

the desired characteristics for normal LOB operations. For example, change the LOB's data storage characteristics to CACHE or NOCACHE LOGGING.

Additional guidelines include: use the NOLOGGING option when writing large amounts of data to the LOB so that when the new version of the LOB data is created, it is not written in the redo log. This improves performance because much less redo log is generated; however, the LOB data must be backed up following a no-logging INSERT operation to ensure the data can be restored.

– CHUNK option

  Set the CHUNK option to the number of blocks of LOB data that are to be accessed at one time, for example, the number of blocks that are to be read or written using the call, OCILobRead( ), OCILobWrite( ), DBMS_LOB.READ( ), or DBMS_LOB.WRITE( ) during one access of the LOB value. Note that the default value for the CHUNK option is one Oracle block and does not vary across systems. For example, if only 1 block of LOB data is accessed at a time, set the CHUNK option to the size of 1 block. For example, if the database block size is 2K, then set the CHUNK option to 2K.

  Set the CHUNKSIZE option slightly larger than the audio, image, or video data size being inserted. Specifying a slightly larger CHUNKSIZE allows for some variation in the actual sizes of the multimedia data and ensures that the benefit is realized. See Example 7–1.

– INITIAL and NEXT parameters

  If you explicitly specify the storage characteristics for the LOB, make sure that the INITIAL and NEXT parameters for the LOB data segment storage are set to a size that is larger than the CHUNK size. For example, if the database block size is 2K and you specify a CHUNK value of 8K, make sure that the INITIAL and NEXT parameters are bigger than 8K and preferably considerably bigger (for example, at least 16K). Also, the INITIAL and NEXT parameters should normally be set to the same value.

– DATABLOCKSIZE

  Oracle manages the storage space in the data files of a database in units called data blocks. The data block is the smallest unit of I/O operation used by a database. You set the data block size for each Oracle database when you create the database. The data block size should be a multiple of the operating system's block size within the maximum (port-specific) limit to avoid unnecessary I/O operations.

For reading and writing LOB data, set the DATABLOCKSIZE parameter to the size of the LOB data you are going to bulk load or to some fractional equivalent. For example, if you are attempting to optimize the LOB storage of 20KB images, set the DATABLOCKSIZE parameter to 10KB or 20KB to move more LOB data in each I/O operation.

– ENABLE | DISABLE STORAGE IN ROW

You use the ENABLE | DISABLE STORAGE IN ROW clause to indicate whether the LOB should be stored inline (that is, in the row) or out of line. You may not alter this specification once you have made it: if you ENABLE STORAGE IN ROW, you cannot alter it to DISABLE STORAGE IN ROW and the reverse. The default is ENABLE STORAGE IN ROW.

The maximum amount of LOB data that will be stored in the row is the maximum VARCHAR size (4000). Note that this includes the control information as well as the LOB value. If the user indicates that the LOB should be stored in the row, once the LOB value and control information are larger than 4000, the LOB value is automatically moved out of the row.

This suggests the following guideline: If the LOB is small (that is, less than 4000 bytes), then storing the LOB data out of line will decrease performance. However, storing the LOB in the row increases the size of the row. This has a detrimental impact on performance if you are doing a lot of base table processing, such as full table scans, multiple row accesses (range scans), or many UPDATE or SELECT statements to columns other than the LOB columns. If you do not expect the LOB data to be less than 4000 bytes, that is, if all LOBs are big, then the default is the best choice because:

(a) The LOB data is automatically moved out of line once it gets bigger than 4000 bytes.

(b) Performance is slightly better because you can still store some control information in the row even after you move the LOB data out of the row.

Example 7–1 assumes that you have already issued a CONNECT statement as some suitably privileged user. This example creates a separate tablespace called MONTANA that is used to store the LOB data for the image column. Ideally, this tablespace would be located on its own high-speed storage device to reduce contention. Other image attributes and the imageID column are stored in the default tablespace. The initial allocation allows 100MB of storage space. The images to be inserted are about 20KB in size. To improve insert performance, NOCACHE and NOLOGGING options are specified along with a CHUNK size of 21KB.

Example 7–1 Creating a Separate Tablespace to Store LOB Data

```
SVRMGR> CREATE TABLESPACE MONTANA DATAFILE 'montana.tbs' SIZE 400M;
Statement processed.
SVRMGR> CREATE TABLE images (image ORDSYS.ORDImage, imageID INTEGER)
     LOB (image.source.localData) STORE AS
            (
            tablespace MONTANA
            STORAGE (
                   INITIAL 100M
                   NEXT 100M
                   )
            CHUNK 21K
            NOCACHE NOLOGGING
            );
```

LOB Buffering

Use LOB buffering, if you need to read or write small pieces of LOB data repeatedly to specific regions of the LOB on the client. Typically, Oracle8*i* options, Web servers, and other client-based applications may need to buffer the contents of one or more LOBs in the client's address space. Using LOB buffering, you can use up to 512K bytes of buffered access. The advantages of LOB buffering include:

- Allowing deferred write operations to the server. Flushing several write operations in the LOBs buffer to the server reduces the number of network roundtrips from the client application to the server, thereby improving overall LOB update performance.
- Reducing the overall number of LOB update operations on the server reduces the number of LOB versions and amount of logging, which improves overall LOB performance and disk space usage.

See *Oracle8i Application Developer's Guide - Large Objects (LOBs)* for further considerations and the use of LOB buffering.

Parallelization

Parallelization is the parallel execution of a single SQL statement using multiple processes. Parallel execution can dramatically improve performance for data-intensive operations associated with decision-support applications or very large database environments such as multimedia databases. Parallel execution is useful for operations accessing large amounts of data and can improve processing for bulk insert, update, and delete operations and processing for objects and data types, such as LOBs. The Oracle database server can use parallel execution for a number of

operations, including PL/SQL functions called from SQL, split partition, update, delete, insert...select, and many other operations. See the *Oracle8i Concepts* manual for a complete list of SQL operations that benefit from parallelization. See the *Oracle8i SQL Reference* manual for examples.

You can use parallelization features for bulk loading of LOB data. See the *Oracle8i Tuning* manual for more information.

7.2 Other Bulk Loading Methods

There are a number of other bulk loading methods available for loading FILE data into LOBs. These include:

- Using the SQL*Loader utility
- Using the OCILobLoadFromFile( ) Relational Function
- Using the DBMS_LOB.LOADFROMFILE( ) Procedure in the DBMS_LOB package

Using the SQL Loader Utility

For Oracle8*i*, you can use SQL*Loader to bulk load objects, collections, and LOBs. This release of SQL*Loader supports loading of the following two object types:

- Column objects
- Row objects

Supported collection types include:

- Nested tables
- Varrays

Supported LOB types include four types of LOBs:

- BLOB: a LOB containing unstructured binary data
- CLOB: a LOB containing single-byte character data
- NCLOB: a LOB containing fixed-size characters from a national character set
- BFILE: a BLOB stored outside of the database tablespaces in a server-side operating system file

See the *Oracle8i Utilities* manual for more information on SQL*Loader and *Oracle8i Application Developer's Guide - Fundamentals* for more information on LOBs.

Using the OCILobLoadFromFile( ) Relational Function

The Oracle Call Interface (OCI) is an application programming interface (API) that allows you to manipulate data and schemas in an Oracle database using a host programming language, such as C.

The OCI relational function, OCILobLoadFromFile( ), loads or copies all or a portion of a FILE into an internal LOB as specified. The data is copied from the source FILE to the destination internal LOB (BLOB/CLOB). No character set conversions are performed when copying the FILE data to a CLOB/NCLOB. Also, when binary data is loaded into a BLOB, no character set conversions are performed. Therefore, the FILE data must already be in the same character set as the LOB in the database. No error checking is performed to verify this.

See *Oracle Call Interface Programmer's Guide* for more information.

Using the DBMS_LOB.LOADFROMFILE( ) Procedure in the DBMS_LOB Package

The DBMS_LOB package provides subprograms to operate on BLOBs, CLOBs, NCLOBs, BFILEs, and temporary LOBs. You can use the DBMS_LOB package for access and manipulation of specific parts of a LOB, as well as complete LOBs. DBMS_LOB can read as well as modify BLOBs, CLOBs, and NCLOBs, and provides read-only operations for BFILEs. The main bulk of the LOB operations are provided by this package.

The DBMS_LOB.LOADFROMFILE( ) procedure copies all, or a part of, a source-external LOB (BFILE) to a destination-internal LOB.

You can specify the offsets for both the source and destination LOBs, and the number of bytes to copy from the source BFILE. The amount and src_offset, because they refer to the BFILE, are in terms of bytes, and the destination offset is either in bytes or characters for BLOBs and CLOBs respectively.

The input BFILE must have been opened prior to using this procedure. No character set conversions are performed implicitly when binary BFILE data is loaded into a CLOB. The BFILE data must already be in the same character set as the CLOB in the database. No error checking is performed to verify this.

See *Oracle8i Application Developer's Reference - Packages* for more information.

7.3 User Guidelines for Best Performance Practices

The following guidelines can be used to help you achieve the best performance when working with LOBs:

- Because LOBs are big, you can attain the best performance by reading and writing large chunks of a LOB value at a time. This helps in several respects:
  - If accessing the LOB from the client side and the client is at a different node than the server, large read/write operations reduce network overhead.
  - If using the NOCACHE option, each small read/write operation incurs an I/O impact. Reading/writing large quantities of data reduces the I/O impact.
  - Writing to the LOB creates a new version of the LOB chunk. Therefore, writing small amounts at a time will incur the cost of a new version for each small write operation. If logging is on, the chunk is also stored in the redo log.
- If you need to read/write small pieces of LOB data on the client, use LOB buffering -- see OCILobEnableBuffering( ), OCILobDisableBuffering( ), OCILobFlushBuffer( ), OCILobWrite( ), OCILobRead( ). Basically, turn on LOB buffering before reading/writing small pieces of LOB data.
- Use OCILobWrite( ) and OCILobRead( ) with a callback so data is streamed to/from the LOB. Ensure that the length of the entire write operation is set in the 'amount' parameter on input. Whenever possible, read and write in multiples of the LOB chunk size.
- Use a checkout/checkin model for LOBs. LOBs are optimized for the following:
  - Updating LOB data: SQL UPDATE operations, which replaces the entire LOB value.
  - Copying the entire LOB data to the client, modifying the LOB data on the client side, and copying the entire LOB data back to the database. This can be done using OCILobRead( ) and OCILobWrite( ) with streaming.

See *Oracle8i Application Developer's Guide - Large Objects (LOBs)* for more information.

7.4 Improving Multimedia LOB Data Retrieval and Update Performance

Once the LOB data is stored in the database, a modified strategy must be used to improve the performance of retrieving and updating the LOB data compared to the insertion strategy described in Section 7.1. The following guidelines should be considered:

- Use the CACHE option on LOBs if the same LOB data is to be accessed frequently by other users.

- Increase the number of buffers if you are going to use the CACHE option.
- Have enough buffers to hold the object. Using a small number of buffers for large objects is not good. Set the db_block_buffers parameter to a value that you know will hold the object.
- Ensure that your redo log files are much larger; otherwise, you can spend lots of time waiting for log switches, especially if you are making lots of updates to your LOB data.
- Ensure that you use a larger page size (db_block_size), especially if the majority of the data in the database is LOB data.

7.5 Setting the Maximum Number of Open BFILEs

A limited number of BFILES can be open simultaneously per session. The default value is 10 files. Set the initialization parameter, SESSION_MAX_OPEN_FILES, to a higher value in the init.ora file if you need to have more BFILEs open simultaneously.

7.6 Using LOBs in Table Partitions

Because you can partition tables containing LOB columns, LOB segments can be spread between several tablespaces to:

- Balance I/O load
- Make backup and recovery operations more manageable
- Make LOB maintenance easier

LOB data can be partitioned to improve I/O problems to better balance the I/O load across the data files of the tablespace containing the LOB data. You can allocate data storage across devices to further improve performance in a practice known as striping. This permits multiple processes to access different portions of the table concurrently without disk contention.

LOB data can be partitioned to tune database backup and recovery operations to make more efficient use of resources needed to perform these operations. For example, having two or more tablespaces that are partitioned lets you perform partial database backup and recovery operations on specific data files as needed.

Similarly, tablespaces with LOBs can be partitioned for easy maintenance of the LOB data by logically grouping LOB data together into smaller partitions that are grouped by date, by subject, by category, and so forth. This makes it easier to add, merge, split, or delete partitions as needed, based on your application.

See *Oracle8i Application Developer's Guide - Large Objects (LOBs)* for examples and further discussion of each of these topics. See the *Oracle8i SQL Reference* manual for examples, specifically the CREATE TABLE statement and the Partitioned Table with LOB Columns example.

A

Audio File and Compression Formats

A.1 Supported Audio File and Compression Formats

The following tables describe the file and compression formats and other audio features supported by *inter*Media audio.

To use these tables, find the data format you are interested in, and then determine the supported formats. For example, Table A–1 shows that *inter*Media audio supports AIFF format for single channel, stereo, 8-bit and 16-bit samples, linear PCM encoding, and uncompressed format.

Table A–1 AIFF Data Format

Format	Audio Feature
AIFF	Single channel
Format ID 'AIFF'	Stereo
File Format: 'AIFF'	8-bit samples
File Ext: .aff	16-bit samples
	Linear PCM encoding
	Compression Format
	Uncompressed

Table A–2 AIFF-C Data Format

Format	Audio Feature
AIFF-C	Single channel
Format ID 'AIFC'	Stereo
File Format: 'AIFC'	8-bit samples
File Ext: .afc	16-bit samples
	Compression Format
Choose one of these compression formats:	Uncompressed
	ADPCM (2:1) compression

Table A–3 AU Data Format

Format	Audio Feature
AU	Single channel
Format ID 'AUFF'	Stereo
File Format: 'AUFF'	8-bit samples
File Ext: .au	16-bit samples
	Mu-law encoding
	Linear PCM encoding
	Compression Format
	Uncompressed

Table A–4 WAV Data Format

Format	Audio Feature
WAV	Single channel
Format ID 'WAVE'	Stereo
File Format: 'WAVE'	8-bit samples
File Ext: .wav	16-bit samples
	Linear PCM encoding
	Compression Format
Choose one of these compression formats:	Uncompressed
	Microsoft ADPCM

B

Image File and Compression Formats

B.1 Supported Image File and Compression Formats

The following tables describe the image file and compression formats supported by Oracle8*i* *inter*Media.

To use these tables, find the data format in which you are interested, and then determine the supported formats. For example, Table B–1 shows that *inter*Media image supports BMP format in monochrome, for read and write access, and in 32-bit RGB for read access.

Table B–1 BMP Data Format

Format	Pixel Format	Support
BMP	Monochrome	Read / Write
	4-bit LUT	Read
File Format: 'BMPF'	8-bit LUT	Read / Write
File Ext: .bmp	16-bit RGB	Read
Mime: image / bmp	24-bit RGB	Read / Write
	32-bit RGB	Read
	Compression Format	**Support**
Choose one of these compression formats:	Uncompressed	Read / Write
	BMPRLE (for 8-bit LUT)	Read / Write
	Data Description	**Support**
Choose one or more of these content formats:	Inverse DIB	Read
	OS / 2 format	Read

Table B–2 CALS Raster Data Format

Format	Pixel Format	Support
CALS Raster File Format: 'CALS' File Ext: .cal Mime: image/x-ora-cals	Monochrome	Read/Write
	Compression Format	**Support**
	FAX4 (CCITT G4)	Read/Write
	Data Description	**Support**
	NA	NA

Table B–3 GIF Data Format

Format	Pixel Format	Support
GIF File Format: 'GIFF' File Ext: .gif Mime: image/gif NOTE: *inter*Media image has limited support for animated GIF images; there is setProperties( ) support; however, processing using the process( ) and processCopy( ) (or Analyze) methods is not supported.	Monochrome 8-bit LUT	Read Read/Write
	Compression Format	**Support**
	GIFLZW (LZW)	Read/Write
	Data Description	**Support**
	NA	NA

Table B–4 JFIF Data Format

Format	Pixel Format	Support
JFIF	8-bit grayscale	Read / Write
	24-bit RGB	Read / Write
File Format: 'JFIF'		
File Ext: .jpg		
Mime: image / jpeg		
	Compression	**Support**
	JPEG	Read / Write
	Data Description	**Support**
	NA	NA

Table B–5 PCX Data Format

Format	Pixel Format	Support
PCX v 5	Monochrome	Read
	2-bit LUT	Read
File Format: 'PCXF'	4-bit LUT	Read
File Ext: .pcx	8-bit LUT	Read
Mime: image / x-ora-pcxf	1-bit RGB	Read
	2-bit RGB	Read
	4-bit RGB	Read
	8-bit RGB	Read
	24-bit RGB	Read
	Compression Format	**Support**
	RLE	Read
	Data Description	**Support**
	NA	NA

Table B–6 PICT Data Format

Format	Pixel Format	Support
PICT v. 1 & 2	Monochrome	Read / Write
	2-bit LUT	Read
File Format: 'PICT'	4-bit LUT	Read
File Ext: .pct	8-bit LUT	Read / Write
Mime: image / pict	16-bit RGB	Read
	24-bit RGB	Read / Write
	Compression Format	**Support**
Choose one of these compression formats:	Packbits	Read / Write
	JPEG (8-bit grayscale and RGB)	Read / Write
	Data Description	**Support**
Choose one or more of these content formats:	Vector / object graphics	Not supported

Table B–7 Raw Pixel Data Format

Format	Pixel Format	Support
Raw Pixel	Monochrome	Read / Write
File Format: 'RPIX'	8-bit grayscale	Read / Write
File Ext: .rpx	24-bit RGB	Read / Write
Mime: image / x-ora-rpix		
	Compression Format	**Support**
Choose one of these compression formats:99	Uncompressed	Read / Write 8-bit grayscale and RGB
	FAX3 (CCITT G3)	Read / Write mono-chrome only
	FAX4 (CCITT G4)	Read / Write mono-chrome only

Table B–7 Raw Pixel Data Format (Cont.)

	Data Description	**Support**
	Inverse scanline order	Read / Write
	Reverse pixel order	Read / Write
	BIP, BIL, or BSQ interleave	Read / Write
	Alternative band order	Read / Write
	>3 bands	Read

Table B–8 Sun Raster Data Format

Format	**Pixel Format**	**Support**
Sun Raster	Monochrome	Read / Write
	8-bit grayscale	Read / Write
File Format: 'RASF'	8-bit LUT	Read / Write
File Ext: .ras	24-bit RGB	Read / Write
Mime: image / x-ora-rasf		
	Compression Format	**Support**
Choose one of these compression formats:	Uncompressed	Read / Write
	SUNRLE (RLE)	Read / Write
	Data Description	**Support**
	NA	NA

Table B–9 Targa Data Format

Format	Pixel Format	Support
Targa	8-bit grayscale	Read / Write
	8-bit LUT	Read / Write
File Format: 'TGAF'	16-bit RGB	Read
File Ext: .tga	24-bit RGB	Read / Write
Mime: image / x-ora-tgaf	32-bit RGB	Read
	Compression Format	**Support**
Choose one of these compression formats:	Uncompressed	Read / Write
	TARGARLE (RLE)	Read / Write
	Data Description	**Support**
	NA	NA

Table B–10 TIFF Data Format

Format	Pixel Format	Support
TIFF v.4/5/6	Monochrome	Read / Write
	8-bit grayscale	Read / Write
File Format: 'TIFF'	4 bit LUT	Read
File Ext: .tif	8-bit LUT	Read / Write
Mime: image / tiff	24-bit RGB	Read / Write
	Compression Format	**Support**

Table B–10 TIFF Data Format (Cont.)

Choose one of these compression formats:	Uncompressed	Read / Write
	Packbits	Read / Write
	Huffman	Read / Write
	FAX3 (CCITT G3)	Read / Write
	FAX4 (CCITT G4)	Read / Write
	LZW	Read / Write
	LZWHDIFF	Read / Write
	JPEG (8-bit gray & RGB)	Read / Write
	Data Description	**Support**
Choose one or more of these content formats:	Planar data	Not supported
	Tiled data	Not supported
	Photometric interpretation	Read / Write
	MSB / LSB	Read / Write

C

Image Process( ) and ProcessCopy( ) Operators

This appendix describes the command options, or operators, used in the process( ) and processCopy( ) methods.

The available operators fall into three broad categories, each described in its own section:

- Section C.2, “Image Formatting Operators”
- Section C.3, “Image Processing Operators”
- Section C.4, “Format-Specific Operators”

Section C.1, “Common Concepts” describes the relative order of these operators.

C.1 Common Concepts

This section describes concepts common to all the image operators and the process( ) and processCopy( ) methods.

C.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.

C.1.2 process( ) and processCopy( )

The process( ) and processCopy( ) methods are functionally identical except for the fact that process( ) writes its output into the same BLOB from which it takes its input while processCopy( ) 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.

C.1.3 Operator and Value

All of the process( ) operators appear in the command string in the form <operator> = <value>. No operator takes effect merely by being present in the command string. The right-hand side of the expression is called the **value** of the operator, and determines how the operator will be applied.

C.1.4 Combining Operators

In general, any number of operators can be combined in the command string passed into the process( ) method if the combination makes sense. However, certain operators are supported only if other operators are present or if other conditions are met. For example, the compressionQuality operator is supported only if the compression format of the destination image is JPEG. Other operators require that the source or destination image be a Raw Pixel or foreign image.

The flexibility in combining operators allows a single operation to change the format of an image, reduce or increase the number of colors, compress the data, and cut or scale the resulting image. This is highly preferable to making multiple calls to do each of these operations sequentially.

C.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)

C.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 file format operator is shown in Table 4–1. Appendix B contains basic information about each file format, including its mnemonic (file format), typical file extension, allowable compression and content formats, and other notable features.

The value given to the file format operator is the single most important detail when specifying the output for process( ). This value determines the range of allowable content and compression formats, whether or not compression quality will be useful, and whether or not the format-specific operators will be useful.

If the FileFormat operator is not used in the process( ) command string, *inter*Media image will determine the file format of the source image and use that as the default file format value. If the file format of the source image does not support output, then an error will occur. If the source image is a foreign image, then the output image will be written as Raw Pixel.

C.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.

The supported values for the ContentFormat operator fall into three broad classes, with three additional special values. The actual mnemonics for these values are listed in Table 4–1.

The content formats whose name includes grayscale or greyscale support only shades of gray. The differences between these content formats is how many shades are allowed. The “4bit” formats support 16 shades while the formats with “8bit” support 256 shades of gray. There is no distinction between grayscale and greyscale.

The content formats whose name includes LUT use a color lookup table to support various colors. The “1bitlut” format allows 2 distinct colors, the “2bitlut” format

supports 4 colors, “4bitlut” supports 16 unique colors, and “8bitlut” supports 256 colors.

The content formats whose name includes RGB store the color values directly in the pixel data as a Red, Green, Blue triplet. The number of bits of total RGB data is specified in the format name and individual formats allocate these bits to red, green, and blue in different ways. However, more bits of data allow for finer distinctions between different shades. Not all bits are used by some image formats.

The monochrome content format allows only black and white to be stored, with no gray shades in between. The “raw” and “24bitplanar” formats are not currently supported.

If the ContentFormat operator is not passed to the process( ) method, then *inter*Media image 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.

C.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 but 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 4–1.

All compression formats that include RLE in their mnemonic are run-length encoding compression schemes, and work well only for images that contain large areas of identical color. The PACKBITS compression type is a run-length encoding scheme that originates from the Macintosh system but is supported by other systems. It has limitations that are similar to other run-length encoding compression formats. Formats that contain LZW or HUFFMAN are more complex compression schemes that examine the image for redundant information and are more useful for a broader class of images. FAX3 and FAX4 are the CCITT Group 3 and Group 4 standards for compressing facsimile data and are useful only for monochrome images. All the compression formats mentioned in this paragraph are **lossless** compression schemes, which means that compressing the image does not discard data. An image compressed into a lossless format and then decompressed will look the same as the original image.

The JPEG compression format is a special case. Developed to compress photographic images, the JPEG format is a **lossy** format, which means that it compresses the image typically by discarding unimportant details. Because this format is opti-

mized for compressing photographic and similarly noisy images, it often produces poor results for other image types, such as line art images and images with large areas of similar color. JPEG is the only lossy compression scheme currently supported by *inter*Media image.

If the CompressionFormat operator is not used, then *inter*Media image will attempt to use the compression format of the source image if the source image has the same file format as the destination image. If the content format of the destination has been altered during processing and the new content format does not support the compression format of the source image, then a default compression format will be chosen; often this default is "None" or "No Compression."

If the CompressionFormat operator is not used and the file format of the destination image is different from that of the source image, then a default compression format will be selected depending on the destination image file format. This default compression is often "None" or "No Compression."

C.2.4 CompressionQuality

The **CompressionQuality** operator determines the relative quality of an image compressed with a lossy compression format. This operator has no meaning for lossless compression formats, and therefore is not currently supported for any compression format except JPEG.

The CompressionQuality operator accepts five values, ranging from most compressed image (lowest visual quality) to least compressed image (highest visual quality): MAXCOMPRATIO, HIGHCOMP, MEDCOMP, LOWCOMP, and MAXINTEGRITY. Using the MAXCOMPRATIO value allows the image to be stored in the smallest amount of space but may introduce visible aberrations into the image. Using the MAXINTEGRITY value keeps the resulting image more faithful to the original but will require more space to store.

If the CompressionQuality operator is not supplied and the destination compression format supports compression quality control, the quality will default to MEDCOMP for medium quality.

C.3 Image Processing Operators

The image processing operators supported by *inter*Media image directly change the way the image looks on the display. The operators supported by *inter*Media image represent only a fraction of all possible image processing operations, and are not intended for users performing intricate image analysis.

C.3.1 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.

C.3.2 Scale

The **Scale** operator enlarges or reduces the image by the ratio given as the value for the operator. If the value is greater than 1.0, then the destination image will be scaled up (enlarged). If the value is less than 1.0, then the output will be scaled down (reduced). A scale value of 1.0 has no effect, and is not an error. No scaling is applied to the source image if the Scale operator is not passed to the process( ) method.

There are two scaling techniques used by *inter*Media image. 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.

C.3.3 XScale

The **XScale** operator is similar to the scale operator but only affects the width (x-dimension) of the image. The important difference between XScale and Scale is that with XScale, scaling by sampling is used whenever the image quality is specified to be MAXCOMPRATIO or HIGHCOMP, and is not dependent on whether the image is being scaled up or down.

This operator may be combined with the YScale operator to scale each axis differently. It may not be combined with other scaling operators (Scale, FixedScale, MaxScale).

C.3.4 YScale

The **YScale** operator is similar to the scale operator but only affects the height (y-dimension) of the image. The important difference between YScale and Scale is that with YScale, scaling by sampling is used whenever the image quality is specified to be MAXCOMPRATIO or HIGHCOMP, and is not dependent on whether the image is being scaled up or down.

This operator may be combined with the XScale operator to scale each axis differently. It may not be combined with other scaling operators (Scale, FixedScale, MaxScale).

C.3.5 FixedScale

The **FixedScale** operator provides an alternate method for specifying scaling values. The Scale, XScale, and YScale operators all accept floating-point scaling ratios, while the FixedScale (and MaxScale) operators specify scaling values in **pixels**. This operator is intended to simplify the creation of images with a specific size, such as thumbnail images.

The two integer values supplied to the FixedScale operator are the desired dimensions (width and height) of the destination image. The supplied dimensions may be larger or smaller (or one larger and one smaller) than the dimensions of the source image.

The scaling method used by this operator will be the same as used by the Scale operator in all cases. This operator cannot be combined with other scaling operators.

C.3.6 MaxScale

The **MaxScale** operator is a variant of the FixedScale operator that preserves the aspect ratio (relative width and height) of the source image. MaxScale also accepts two integer dimensions, but these values represent the maximum value of the appropriate dimension after scaling. The final dimension may actually be less than the supplied value.

Like the FixedScale operator, this operator is also intended to simplify the creation of images with a specific size. MaxScale is even better suited to thumbnail image creation than the FixedScale operator because thumbnail images created using MaxScale will have the correct aspect ratios.

The MaxScale operator scales the source image to fit within the dimensions specified while preserving the aspect ratio of the source image. Because the aspect ratio is preserved, only one dimension of the destination image may actually be equal to

the values supplied to the operator. The other dimension may be smaller than or equal to the supplied value. Another way to think of this scaling method is that the source image is scaled by a single scale factor that is as large as possible with the constraint that the destination image fit entirely within the dimensions specified by the MaxScale operator.

If the Cut operator is used in conjunction with the MaxScale operator, then the aspect ratio of the cut window is preserved instead of the aspect ratio of the input image.

The scaling method used by this operator is the same as used by the Scale operator in all cases. This operator cannot be combined with other scaling operators.

C.4 Format-Specific Operators

The following operators are supported only when the destination image file format is Raw Pixel, with the exception of the InputChannels operator, which is supported only when the source image is Raw Pixel or a foreign image. It does not matter if the destination image format is set to Raw Pixel explicitly using the FileFormat operator, or if the Raw Pixel format is selected by *inter*Media image automatically because the source format is Raw Pixel or a foreign image.

C.4.1 ChannelOrder

The **ChannelOrder** operator determines the relative order of the red, green, and blue channels (bands) within the destination Raw Pixel image. The order of the characters R, G, and B within the mnemonic value passed to this operator determine the order of these channels within the output. The header of the Raw Pixel image will be written such that this order is not lost.

For more information about the Raw Pixel file format and the ordering of channels in that format, see Appendix D.

C.4.2 Interleave

The **Interleave** operator controls the layout of the red, green, and blue channels (bands) within the destination Raw Pixel image.The three mnemonic values supported by this operator: BIP, BIL, and BSQ force the output image to be “band interleaved by pixel,” “band interleaved by line,” and “band sequential” respectively.

For more information about the Raw Pixel file format, the interleaving of channels in that format, or the meaning of these interleaving values, see Appendix D.

C.4.3 PixelOrder

The **PixelOrder** operator controls the direction of pixels within a scanline in a Raw Pixel Image. The value Normal indicates that the leftmost pixel of a scanline will appear first in the image data stream. The value Reverse causes the rightmost pixel of the scanline to appear first.

For more information about the Raw Pixel file format and pixel ordering, see Appendix D.

C.4.4 ScanlineOrder

The **ScanlineOrder** operator controls the order of scanlines within a Raw Pixel image. The value Normal indicates that the top display scanline will appear first in the image data stream. The value Inverse causes the bottom scanline to appear first.

For more information about the Raw Pixel file format and scanline ordering, see Appendix D.

C.4.5 InputChannels

As stated in Section C.4, the **InputChannels** operator is supported only when the source image is in Raw Pixel format or if the source is a foreign image.

The InputChannels operator assigns individual bands from a multiband image to be the red, green, and blue channels for later image processing. Any band within the source image can be assigned to any channel. If desired, only a single band may be specified and the selected band will be used as the gray channel, resulting in a gray-scale output image. The first band in the image is number 1, and the band numbers passed to the Input Channels operator must be greater than or equal to one, and less than or equal to the total number of bands in the source image. Only the bands selected the by InputChannels operator are written to the output. Other bands are not transferred, even if the output image is in Raw Pixel format.

It should be noted that every Raw Pixel or foreign image has these input channel assignments written into its header block, but that this operator overrides those default assignments.

For more information about the Raw Pixel file format and input channels, see Appendix D.

D

Image Raw Pixel Format

This appendix describes the Oracle Raw Pixel image format and is intended for developers and advanced users who wish to use the Raw Pixel format to import unsupported image formats into *inter*Media image, or as a means to directly access the pixel data in an image.

Much of this appendix is also applicable to foreign images.

D.1 Raw Pixel Introduction

*inter*Media image supports many popular image formats suitable for storing artwork, photographs, and other images in an efficient, compressed way, and provides the ability to convert between these formats. However, most of these formats are proprietary to at least some degree, and the format of their content is often widely variable and not suited for easy access to the pixel data of the image.

The Raw Pixel format is useful for applications that need direct access to the pixel data without the burden of the complex computations required to determine the location of pixels within a compressed data stream. This simplifies reading the image for applications that are performing pixel-oriented image processing, such as filtering and edge detection. This format is even more useful to applications that need to write data back to the image. Because changing even a single pixel in a compressed image can have implications for the entire image stream, providing an uncompressed format enables applications to write pixel data directly, and later compress the image with a single process( ) command.

This format is also useful to users who have data in a format not directly supported by *inter*Media image, but is in a simple, uncompressed format. These users can prepend a Raw Pixel identifier and header onto their data and import it into *inter*Media image. For users who only need to read these images (such as for import or conversion), this capability is built into *inter*Media image as “Foreign Image Sup-

port". How this capability is related to the Raw Pixel format is described in Section D.10.

In addition to supporting image types not already built into *inter*Media image, the Raw Pixel format also permits the interpretation of N-band imagery, such as satellite images. Using Raw Pixel, one or three bands of an N-band image may be selected during conversion to another image format, allowing easy visualization within programs that do not otherwise support N-band images. Note that images written with the Raw Pixel format still may only have one or three bands.

The current version of the Raw Pixel format is "1.0". This appendix is applicable to Raw Pixel images of this version only, as the particulars of the format may change with other versions.

D.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 zero or more bytes, followed by pixel data.

It is worth noting that Raw Pixel images are never color-mapped, and therefore do not contain color lookup tables.

The Raw Pixel header consists of the "Image Identifier" and the "Image Header". The Image Header is actually composed of several fields.

Note that the first byte in the image is actually offset '0'. All integer fields are unsigned and stored in big endian byte order.

Name	Byte(s)	Description
Image Identifier	0:3	4-byte character array containing ASCII values for RPIX. This array identifies the image as a Raw Pixel image.
Image Header Length	4:7	Length of this header in bytes, excluding the identifier field. The value of this field may be increased to create a gap between the header fields and the pixel data in the image.
Major Version	8	Major version number of the Raw Pixel format used in the image.
Minor Version	9	Minor version number of the Raw Pixel format used in the image.
Image Width	10:13	Width of the image in pixels.

Image Height	14:17	Height of the image in pixels.
Compression Type	18	Compression type of the image: None, CCITT FAX Group 3, or CCITT FAX Group 4.
Pixel Order	19	Pixel order of the image: Normal or Reverse.
Scanline Order	20	Scanline order of the image: Normal or Inverse.
Interleave	21	Interleave type of the image: BIP, BIL, or BSQ.
Number of Bands	22	Number of bands in the image. Must be in the range 1 to 255.
Red Channel Number	23	The band number of the channel to use as a default for red. This field is the gray channel number if the image is grayscale.
Green Channel Number	24	The band number of the channel to use as a default for green. This field is zero if the image is grayscale.
Blue Channel Number	25	The band number of the channel to use as a default for blue. This field is zero if the image is grayscale.
Reserved Area	26:33	Not currently used. All bytes ***must*** be zero.

D.3 Raw Pixel Header Field Descriptions

This section describes the fields of the Raw Pixel Header in greater detail.

Image Identifier

Occupying the first 4 bytes of a Raw Pixel image, the identifier string must always be set to the ASCII values “RPIX” (hex 52 50 49 58). These characters identify the image as being encoded in RPIX format.

This string is currently independent of the Raw Pixel version.

Image Header Length

The Raw Pixel reader uses the value stored in this field to find the start of the pixel data section within a Raw Pixel image. To find the offset of the pixel data in the image, the reader adds the length of the image identifier (always ‘4’) to the value in

the image header length field. Thus, for Raw Pixel 1.0 images with no post-header gap, the pixel data starts at offset 34.

For Raw Pixel version 1.0 images, this field normally contains the integer value '30', which is the length of the Raw Pixel image header (not including the image identifier). However, the Raw Pixel format allows this field to contain any value equal to or greater than 30. Any information in the space between the end of the header data and the start of the pixel data specified by this header length is ignored by the Raw Pixel reader. This is useful for users who wish to prepend a Raw Pixel header onto an existing image whose pixel data area is compatible with Raw Pixel. 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 Raw Pixel). This field is stored in big endian byte order.

Major Version

A single-byte integer containing the major version number of the Raw Pixel format version used to encode the image. The current Raw Pixel version is "1.0", therefore this field is '1'.

Minor Version

A single-byte integer containing the minor version number of the Raw Pixel format version used to encode the image. The current Raw Pixel version is "1.0", therefore this field is '0'.

Image Width

The width (x-dimension) of the image in pixels.

Although this field is capable of storing an image dimension in excess of 4 billion pixels, limitations within *inter*Media image require that this field be in the range 1<= width <= 32767. This field is stored in big endian byte order.

Image Height

The height (y-dimension) of the image in pixels.

Although this field is capable of storing an image dimension in excess of 4 billion pixels, limitations within *inter*Media image require that this field be in the range 1 <= height <= 32767. This field is stored in big endian byte order.

Compression Type

This field contains the compression type of the Raw Pixel image. As of version 1.0, this field may contain the following values:

Value	Name	Compression
1	NONE	No compression
2	FAX3	CCITT Group 3 compression
3	FAX4	CCITT Group 4 compression

For grayscale, RGB, and N-band images, the image is always uncompressed, and only a value of 0 is valid. If the compression type is value 1 or 2, then the image is presumed to be monochrome. In this case the image is presumed to contain only a single band, and must specify normal pixel order, normal scanline order, and BIP interleave.

Pixel Order

This field describes the pixel order within the Raw Pixel image. Normally, pixels in a scanline are ordered from left to right, along the traditional positive x-axis. However, some applications require that scanlines be ordered from right to left.

This field may contain the following values:

Value	Name	Pixel Order
1	NORMAL	Leftmost pixel first
2	REVERSE	Rightmost pixel first

This field cannot contain 0, as this indicates an unspecified pixel order; this would mean the image could not be interpreted. For images with CCITT G3 and G4 compression types, this field must contain the value ‘1’.

Scanline Order

This field describes the scanline order within the Raw Pixel image. Normally, scanlines in an image are ordered from top to bottom. However, some applications require that scanlines are ordered from bottom to top.

This field may contain the following values:

Value	Name	Scanline Order
1	NORMAL	Topmost scanline first
2	INVERSE	Bottommost scanline first

This field cannot contain 0, as this indicates an unspecified scanline order; this would mean the image could not be interpreted. For images with CCITT G3 and G4 compression types, this field must contain the value 1.

Interleave

This field describes the interleaving of the various bands within a Raw Pixel image. For more information on the meaning of the various interleave options, see Section D.5.3.

This field may contain the following values:

Value	Name	Interleave
1	BIP	Band Interleave by Pixel, or “chunky”
2	BIL	Band Interleave by Line
3	BSQ	Band SeQuential, or “planar”

This field cannot contain 0, as this indicates an unspecified interleave; this would mean the image could not be interpreted. For images with CCITT G3 and G4 compression types, this field must contain the value 1.

Number of Bands

This field contains the number of bands or planes in the image, and must be in the range 1 <= number of bands <= 255. This field may not contain the value 0.

For CCITT images, this field must contain the value 1.

Red Channel Number

This field contains the number of the band that is to be used as the red channel during image conversion operations. This may be used to change the interpretation of a normal RGB image, or to specify a default band to be used as red in an N-band image. This default may be overridden using the inputChannels operator in the process( ) or processCopy( ) methods.

If the image has only one band, or only one band from an N-band image should be selected for display, then the band number should be encoded as the red channel. In this case, the green and blue channels should be set to 0.

This field may not contain the value 0; only values in the range (1 <= red <= number of bands) may be specified.

Green Channel Number

This field contains the number of the band that is to be used as the green channel during image conversion operations. This may be used to change the interpretation of a normal RGB image, or to specify a default band to be used as green in an N-band image. This default may be overridden using the inputChannels operator in the process( ) or processCopy( ) method.

If the image has only one band, or only one band from an N-band image should be selected for display, then the band number should be encoded as the red channel. In this case, the green and blue channels should be set to 0.

This field may contain values in the range 0 <= green <= number of bands.

Blue Channel Number

This field contains the number of the band that is to be used as the blue channel during image conversion operations. This may be used to change the interpretation of a normal RGB image, or to specify a default band to be used as blue in an N-band image. This default may be overridden using the inputChannels operator in the process( ) or processCopy( ) method.

If the image has only one band, or only one band from an N-band image should be selected for display, then the band number should be encoded as the red channel. In this case, the green and blue channels should be set to 0.

This field may contain values in the range 0 <= blue <= number of bands.

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 zero. Failure to do so will create undefined results.

D.4 Raw Pixel Post-Header Gap

Apart from the image identifier and the image header, Raw Pixel version 1.0 images contain an optional post-header gap, which precedes the actual pixel data. Unlike the reserved area of the image header, the bytes in this gap can contain any values

you want. This is useful to store additional metadata about the image, which in some cases may be the actual image header from another file format.

However, because there is no standard for the information stored in this gap, care must be taken if metadata is stored in this area as other users may interpret this data differently. It is also worth noting that when a Raw Pixel image is processed, information stored in this gap is not copied to the destination image. In the case of the process( ) method, which writes its output to the same location as the input, the source information will be lost unless the transaction in which the processing took place is rolled back.

D.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 lay-out of the bitmap data.

For images using CCITT compression, the bitmap data area stores the raw CCITT stream with no additional header. The rest of this section applies only to uncompressed images.

Bitmap data in a Raw Pixel image is stored as 8-bit per plane, per pixel, direct color, packed data. There is no pixel, scanline, or band blocking or padding. Scanlines may be presented in the image as either topmost first, or bottommost first. Within a scanline, pixels may be ordered leftmost first, or rightmost first. All these options are affected by interleaving in a relatively straightforward way; see the sections that follow for examples.

D.5.1 Scanline Ordering

On the screen, an image may look like the following:

```
1111111111...
2222222222...
3333333333...
4444444444...
```

Each digit represents a single pixel; the value of the digit is the scanline that the pixel is on.

Generally the scanline that forms the upper or topmost row of pixels is stored in the image data stream before lower scanlines. The preceding image would appear as follows in the bitmap data stream:

```
...1111111111...2222222222...3333333333...4444444444...
```

Note that the first scanline appears earlier than the remaining scanlines. The Raw Pixel format refers to this scanline ordering as normal.

However, some applications prefer that the bottommost scanline appear in the data stream first:

...4444444444...3333333333...2222222222...1111111111...
The Raw Pixel format refers to this scanline ordering as inverse.

D.5.2 Pixel Ordering

On the screen, a scanline of an image may look like the following:

...123456789...

Each digit represents a single pixel; the value of the digit is the column that the pixel is on.

Generally the data that forms the leftmost pixels is stored in the image data stream before pixels toward the right. The preceding scanline would appear as follows in the bitmap data stream:

...123456789...
Note that the left pixel appears earlier than the remaining pixels. The Raw Pixel format refers to this pixel ordering as normal.

However, some applications prefer that the rightmost pixel appear in the data stream first:

...987654321...
The Raw Pixel format refers to this pixel ordering as reverse.

D.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)
...
```

D.5.4 N-Band Data

The Raw Pixel format supports up to 255 bands of data in an image. The relative location of these bands of data in the image is described in Section D.5.3, which gives examples of interleaving for 3 bands of data.

In the case of a single band of data, there is no interleaving; all three schemes are equivalent. Examples of interleaving other numbers of bands are given in the following table. All images have three scanlines and four columns. Each band of each pixel is represented by a single-digit band number. Normal text numbers in *italic* represent the second scanline of the image, and numbers in **boldface** represent the third scanline of the image.

Bands	BIP	BIL	BSQ
2	12121212 *12121212* **12121212**	11112222 *11112222* **11112222**	1111*1111***1111** 2222*2222***2222**
4	1234123412341234 *1234123412341234* 1**234123412341234**	1111222233334444 *1111222233334444* **1111222233334444**	1111*1111***1111** 2222*2222***2222** 3333*3333***3333** 4444*4444***4444**
5	12345123451234512345 *12345123451234512345* **12345123451234512345**	11112222333344445555 *11112222333344445555* **11112222333344445555**	1111*1111***1111** 2222*2222***2222** 3333*3333***3333** 4444*4444***4444** 5555*5555***5555**

D.6 Raw Pixel Header “C” 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/Planar) */

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 */
};
```

D.7 Raw Pixel Header “C” Constants

The following C language constants define the values used in the Raw Pixel header.

```
#define RPIX_IDENTIFIER "RPIX"

#define RPIX_HEADERLENGTH 30

#define RPIX_MAJOR_VERSION 1
#define RPIX_MINOR_VERSION 0

#define RPIX_COMPRESSION_UNDEFINED 0
#define RPIX_COMPRESSION_NONE 1
#define RPIX_COMPRESSION_CCITT_FAX_G3 2
#define RPIX_COMPRESSION_CCITT_FAX_G4 3
#define RPIX_COMPRESSION_DEFAULT RPIX_COMPRESSION_NONE

#define RPIX_PIXEL_ORDER_UNDEFINED 0
#define RPIX_PIXEL_ORDER_NORMAL 1
#define RPIX_PIXEL_ORDER_REVERSE 2
#define RPIX_PIXEL_ORDER_DEFAULT RPIX_PIXEL_ORDER_NORMAL

#define RPIX_SCANLINE_ORDER_UNDEFINED 0
#define RPIX_SCANLINE_ORDER_NORMAL 1
```

```
#define RPIX_SCANLINE_ORDER_INVERSE 2
#define RPIX_SCANLINE_ORDER_DEFAULT RPIX_SCANLINE_ORDER_NORMAL

#define RPIX_INTERLEAVING_UNDEFINED 0
#define RPIX_INTERLEAVING_BIP 1
#define RPIX_INTERLEAVING_BIL 2
#define RPIX_INTERLEAVING_BSQ 3
#define RPIX_INTERLEAVING_DEFAULT RPIX_INTERLEAVING_BIP

#define RPIX_CHANNEL_UNDEFINED 0
```

Note that the various macros for the UNDEFINED values are meant to be illustrative and not necessarily used, except for "RPIX_CHANNEL_UNDEFINED" which is used for the green and blue channels of single band images.

D.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;
```

D.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 as even grayscale, as the unused data bits combined with the very high resolution used in these images would use excessive disk space.

Raw Pixels images using CCITT compression are treated as normal Raw Pixel images, with the following restrictions:

- The “compression type” field must contain the value 1 or 2 as outlined in Section D.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 "blue channel number" fields must contain the value 0 (no band).

In addition to these restrictions, applications which attempt to access pixel data directly will need to understand how to read and write the CCITT formatted data.

D.10 Foreign Image Support and the Raw Pixel Format

*inter*Media image 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 so large and continually changing. Instead, there are some simple guidelines to determine if an image can be read using the foreign image support in *inter*Media image. These rules are summarized in the following sections.

Header

Foreign images may have any header (or no header), in any format, as long as its length does not exceed 4,294,967,265 bytes. As has been noted before, all information in this header will be ignored.

Image Width

Foreign images may be up to 32,767 pixels wide.

Image Height

Foreign images may be up to 32,767 pixels high.

Compression Type

Foreign images must be uncompressed or compressed using CCITT Fax Group 3 or Fax Group 4. Other compression schemes, such as run-length encoding, are not currently supported.

Pixel Order

Foreign images may store pixels from left-to-right or right-to-left. Other pixel ordering schemes, such as boustrophedonic ordering, are not currently supported.

Scanline Order

Foreign images may have top-first or bottom-first scanline orders. Scanlines that are adjacent in the image display must be adjacent in the image storage. Some image formats stagger their image scanlines so that, for example, scanlines 1,5,9, and so forth are adjacent, and then 2,6,10 are also adjacent. This is not currently supported.

Interleave

Foreign images must use BIP, BIL, or BSQ interleaving. Other arrangements of data bands are not allowed, nor may bands have any pixel, scanline, or band-level blocking or padding.

Number of Bands

Foreign images may have up to 255 bands of data. If there are more bands of data, the first 255 can be accessed *if* the interleaving of the image is "band sequential." In this case, the additional bands of data lie past the accessible bands and do not affect the layout of the first 255 bands. Images with other interleaving types may not have more than 255 bands because the additional bands will change the layout of the bitmap data.

Trailer

Foreign images may have an image trailer following the bitmap data, and this trailer may be of arbitrary length. However, such data is completely ignored by *inter*Media image, and there is no method (or need) to specify the presence or length of such a trailer.

If an image with such a trailer is modified with the process( ) or processCopy( ) methods, the resulting image will not contain this trailer. In the case of the processCopy( ) method, the source image will still be intact.

E

Sample Programs

Oracle8*i* *inter*Media includes a number of scripts and sample programs that you can use.

Sample Oracle8*i* *inter*Media scripts and programs are available in the following directories after you install this product:

$ORACLE_HOME/ord/aud/demo/
$ORACLE_HOME/ord/img/demo/
$ORACLE_HOME/ord/vid/demo/

E.1 Sample Audio Scripts

The audio scripts consist of the following files:

- auddemo.sql - audio demonstration (demo) that shows features of the audio object including:
  - Checking *inter*Media objects
  - Creating a sample table with audio in it
  - Inserting NULL rows into the audio table
  - Checking the rows out
  - Checking all the audio attributes directly
  - Checking all the audio attributes by calling methods
  - Installing your own format plug-in using the two files, fplugins.sql and fpluginb.sql described in the next two list items and in Section 2.1.14 on how to extend *inter*Media audio to support a new audio data format
- fplugins.sql - demo format plug-in specification that you can use as a guideline to write any format plug-in you want to support

- fpluginb.sql - demo format plug-in body that you can use as a guideline to write any format plug-in you want to support

See the README.txt file in the *$ORACLE_HOME*/ord/aud/demo directory for requirements and instructions on running this SQL demo.

E.2 Sample Program for Modifying Images or Testing the Image Installation

Once you have installed Oracle8*i* *inter*Media image, you may choose to run the Oracle8*i* *inter*Media image demonstration program. This program can also be used as a test to confirm successful installation.

This section contains the steps required to build and run the *inter*Media image demo.

The *inter*Media image demo files are located in <*ORACLE_HOME*>*/ord/img/*demo, where <*ORACLE_HOME*> is the ORACLE_HOME directory.

E.2.1 Demonstration (Demo) Installation Steps

1. The Oracle8*i* *inter*Media image demo uses the SCOTT/TIGER database user. If this user does not exist, you must create it:

   ```
   % svrmgrl
   SVRMGRL> connect internal;
   SVRMGRL> create user SCOTT identified by tiger;
   SVRMGRL> grant connect,resource to SCOTT;
   ```

2. Create the image demo directory where <*ORACLE_HOME*> is the ORACLE_HOME directory.

   ```
   % svrmgrl
   SVRMGRL> connect internal;
   SVRMGRL> create or replace directory imgdemodir as '<ORACLE_HOME>/ord/img/
   demo';
   ```

   The directory specification is operating system specific. See the installation and configuration guide for your operating system for details on how to properly specify the location.

3. Grant privileges on the directory to PUBLIC.

   ```
   SVRMGRL> grant read on directory imgdemodir to public with grant option;
   ```

4. If needed, make the imgdemo program.

```
% cd <ORACLE_HOME>/ord/img/demo
% make -f demo_ordimg.mk imgdemo
```

The make operation is operating system specific. See the installation and configuration guide for your operating system for details on how to make this program on your operating system.

E.2.2 Running the Demo

The file imgdemo is a sample program that shows how Oracle8*i interM*edia image can be used from within a program. The demo is written in C and uses OCI, Oracle Call Interface, to access the database and exercise Oracle8*i interM*edia image.

The program operates on imgdemo.dat, which is a bitmap (BMP) image in the demo directory. Optionally, you can supply an image file name on the command line, provided the file resides in the same directory as the demo. In either case, once the image has been manipulated by Oracle8*i interM*edia image, the resulting image is written to the file imgdemo.out and can then be viewed with common rendering tools that you supply.

When the demo is run, it deletes and re-creates a table named IMGDEMOTAB in the SCOTT/TIGER schema of the default database. This table is used to hold the demo data. Once the table is created, a reference to the image file is inserted into the table. The data is then loaded into the table and converted to JFIF using the processCopy( ) method of ORDImage.

The image properties are extracted within the database using the setProperties( ) method. An UPDATE command is issued after the setProperties( ) invocation. This is required because the setProperties( ) invocation has only updated a local copy of the type attributes.

Next, the Oracle8*i interM*edia image process( ) method is used to cut and scale the image within the database. This is followed by an update that commits the change. The program cuts a portion of the image 100 pixels wide by 100 pixels high starting from pixel location (100,100). This subimage is scaled to twice its original size and the resulting image is written out to the file system in a file named imgdemo.out.

Upon completion, the demo program leaves the imgdemo.out file in the current directory. It also leaves the table IMGDEMOTAB in the SCOTT/TIGER schema of the database.

Example E–1 Execute the Demo from the Command Line

Execute the demo by typing imgdemo on the command line. Optionally, a different image can be used in the demo by first copying the file to the directory in which the

demo resides and then specifying its file name on the command line as an argument to imgdemo.

Use the following command:

```
$ imgdemo <optional-image-filename>
```

The demo displays a number of messages describing its progress, along with any errors encountered in the event that something was not set up correctly. Expect to see the following messages:

```
Dropping table IMGDEMOTAB...
Creating and populating table IMGDEMOTAB...
Loading data into cartridge...
Modifying image characteristics...
Writing image to file imgdemo.out...
Disconnecting from database...
Logged off and detached from server.
Demo completed successfully.
```

If the program encounters any errors, it is likely that either Oracle8*i* *inter*Media image software has not been installed correctly or the database has not been started. If the program completes successfully, the original image and the resultant image, which has undergone the cutting and scaling described earlier, can be viewed with common image rendering tools.

E.3 Sample Video Scripts

The video scripts consist of the following files:

- viddemo.sql - video demo that shows features of the video object including:
  - Checking *inter*Media objects
  - Creating a sample table with video in it
  - Inserting NULL rows into the video table
  - Checking the rows out
  - Checking all the video attributes directly
  - Checking all the video attributes by calling methods
  - Installing your own format plug-in using the two files, fplugins.sql and fpluginb.sql described in the next two list items and in Section 2.3.12 on how to extend *inter*Media video to support a new video data format

- fplugins.sql - demo format plug-in specification that you can use as a guideline to write any format plug-in you want to support
- fpluginb.sql - demo format plug-in body that you can use as a guideline to write any format plug-in you want to support

See the README.txt file in the *$ORACLE_HOME*/ord/vid/demo directory for requirements and instructions on how to run this SQL demo.

E.4 Java Demo

A Java demo has been provided to help you learn to use both the audio and video client-side Java classes so you can build your own applications. In these two demos, the audio and video object is instantiated at the client side and a number of accessor methods are invoked. The audio Java demo files are located in the *ORACLE_HOME/*ord/aud/demo directory and the video Java demo files are located in the *$ORACLE_HOME*/ord/vid/demo directory. See the README.txt file in each directory for requirements and instructions on how to run each respective Java demo.

F

Reference Information

This appendix describes additional reference information presented elsewhere in this guide as part of the demo program. It describes the following:

- ORDAnnotations type and its methods

F.1 Object Types

Oracle8*i* *inter*Media makes use of the following object types in its demo program:

- ORDAnnotations: supports the storage of annotation information

ORDAnnotations Object Type

The ORDAnnotations object type supports the storage of annotation information.

This object type is defined in the following list:

```
CREATE OR REPLACE TYPE ORDAnnotations
AS OBJECT
(
  -- TYPE ATTRIBUTES
annotations        ORDAnnotationList,
  -- METHOD DECLARATION
MEMBER PROCEDURE setAnnotation(key   IN VARCHAR2,
                               value IN VARCHAR2),
MEMBER FUNCTION getAnnotation(key IN VARCHAR2)
                RETURN VARCHAR2,
PRAGMA RESTRICT_REFERENCES(getAnnotation, WNDS, WNPS, RNPS),

MEMBER PROCEDURE deleteAnnotation(key IN VARCHAR2),
MEMBER PROCEDURE appendToAnnotation(key   IN VARCHAR2,
                                    value IN VARCHAR2),
MEMBER FUNCTION getAnnotationCount RETURN INTEGER,
PRAGMA RESTRICT_REFERENCES(getAnnotationCount, WNDS, WNPS, RNPS),

MEMBER FUNCTION getAnnotationObjByPosition(pos IN INTEGER)
                RETURN ORDAnnotation,
PRAGMA RESTRICT_REFERENCES(getAnnotationObjByPosition, WNDS, WNPS, RNPS),

MEMBER FUNCTION getAnnotationKeyByPosition(pos IN INTEGER)
                RETURN VARCHAR2,
PRAGMA RESTRICT_REFERENCES(getAnnotationKeyByPosition, WNDS, WNPS, RNPS),

MEMBER FUNCTION getAnnotationValueByPosition(pos IN INTEGER)
                RETURN VARCHAR2,
PRAGMA RESTRICT_REFERENCES(getAnnotationValueByPosition, WNDS, WNPS, RNPS),

);
```

where:

- annotations: is an ORDAnnotationList table containing one or more ORD-Annotation objects of which each ORDAnnotation object contains a key defined as a VARCHAR2(256) and a value defined as a VARCHAR2(4000).

F.2 Methods

This section presents reference information on the methods used for annotation manipulation.

The ORDAnnotations methods are described as follows:

ORDAnnotations Methods Associated with the annotations Attribute

- setAnnotation( ): creates the values for the ORDAnnotation object key and value fields.
- getAnnotation( ): returns the value of the value field for the ORDAnnotation object given its key value.
- deleteAnnotation( ): deletes the ORDAnnotation object by its key value.
- appendToAnnotation( ): appends information to the values of the ORDAnnotation object's key and value fields.
- getAnnotationCount: returns the number of ORDAnnotation objects in the ORDAnnotationList table.
- getAnnotationObjByPosition( ): returns the ORDAnnotation object by its position in the ORDAnnotationList table.
- getAnnotationKeyByPosition( ): returns the value of the key field by the ORDAnnotation object's position in the ORDAnnotationList table.
- getAnnotationValueByPosition( ): returns the value of the value field by the ORDAnnotation object's position in the ORDAnnotationList table.

F.2.1 ORDAnnotations Methods Associated with the annotations Attribute

This section presents reference information on the ORDAnnotations methods associated with the annotations attribute.

The methods described in this reference chapter show examples based on a TANNOT table. Refer to the TANNOT table definition that follows when reading through the examples in this chapter:

```
CREATE TABLE TANNOT(n NUMBER, a ORDSYS.ORDAnnotations)
storage (initial 100K next 100K pctincrease 0),
nested table a.annotations store as base_annot;
INSERT INTO TANNOT VALUES(1, ORDSYS.ORDAnnotations(NULL));
```

setAnnotation( ) Method

Format

```
setAnnotation(key   IN VARCHAR2,
              value IN VARCHAR2);
```

Description

Creates or alters the values for the key and value fields for the ORDAnnotation object.

Parameters

key
The value of the key field for the ORDAnnotation object.

value
The value of the value field for the ORDAnnotation object.

Usage

none

Pragmas

none

Exception

none

Example

none

getAnnotation( ) Method

Format

```
getAnnotation(key IN VARCHAR2) RETURN VARCHAR2;
```

Description

Returns the value of the value field for the ORDAnnotation object based on the value of the key field.

Parameters

key
The value of the key field for the ORDAnnotation object.

Usage

none

Pragmas

Pragma RESTRICT_REFERENCES(getAnnotation, WNDS, WNPS, RNPS)

Exception

If the key cannot be found based on its value, then calling this method raises a KEY_NOT_FOUND exception.

If the value of the annotations attribute is empty (NULL), then calling this method raises a NULL_ANNOTATION_LIST exception.

Example

Return the value of the key and value fields for the first ORDAnnotation object in the ORDAnnotationList table of the ORDAnnotations object:

```
DECLARE
  mya ORDSYS.ORDAnnotations;
BEGIN
 SELECT T.a INTO mya FROM TANNOT T WHERE N = 1;
 DBMS_OUTPUT.PUT_LINE('Annotation for key: '||mya.getAnnotation('key'));
 DBMS_OUTPUT.PUT_LINE('Annotation for value:'||mya.getAnnotation('note1'));
  EXCEPTION
```

```
  WHEN ORDSYS.ORDAnnotationsExceptions.KEY_NOT_FOUND THEN
    DBMS_OUTPUT.PUT_LINE('key does not exist in annotations');
END;
```

deleteAnnotation( ) Method

Format

```
deleteAnnotation(key IN VARCHAR2);
```

Description

Deletes the ORDAnnotation object based on the value of its key field.

Parameters

key
The value of the key field for the ORDAnnotation object.

Usage

none

Pragmas

none

Exception

If the key cannot be found based on its value, then calling this method raises a KEY_NOT_FOUND exception.

Example

none

appendToAnnotation( ) Method

Format

```
appendToAnnotation(key   IN VARCHAR2,
                   value IN VARCHAR2);
```

Description

Appends the specified values for the key and value fields to the ORDAnnotation object's current values.

Parameters

key
The appended value of the key field for the ORDAnnotation object.

value
The appended value of the value field for the ORDAnnotation object.

Usage

none

Pragmas

none

Exception

If the key cannot be found based on its value, then calling this method raises a KEY_NOT_FOUND exception.

If the value of the annotations attribute is empty (NULL), then calling this method raises a NULL_ANNOTATION_LIST exception.

Example

none

getAnnotationCount Method

Format

```
getAnnotationCount RETURN INTEGER;
```

Description

Returns the number of ORDAnnotation objects in the ORDAnnotationList table.

Parameters

none

Usage

none

Pragmas

Pragma RESTRICT_REFERENCES(getAnnotationCount, WNDS, WNPS, RNPS)

Exception

none

Example

Return the ORDAnnotation object count:

```
DECLARE
  mya ORDSYS.ORDAnnotations;
BEGIN
 SELECT T.a INTO mya FROM TANNOT T WHERE N = 1;
 DBMS_OUTPUT.PUT_LINE(TO_CHAR(mya.getAnnotationCount));
END;
```

getAnnotationObjByPosition( ) Method

Format

```
getAnnotationObjByPosition(pos IN INTEGER) RETURN ORDAnnotation;
```

Description

Returns the ORDAnnotation object by its position in the ORDAnnotationList table.

Parameters

pos
The ORDAnnotation object's position in the ORDAnnotationList table.

Usage

none

Pragmas

Pragma RESTRICT_REFERENCES(getAnnotationObjByPosition, WNDS, WNPS, RNPS)

Exception

If the value of pos is greater than the number of ORDAnnotation objects in the ORDAnnotationList table, then calling this method raises a POSITION_OUT_OF_RANGE exception.

Example

Return the ORDAnnotation object by its position in the ORDAnnotationList table:

```
DECLARE
  mya ORDSYS.ORDAnnotations;
  obj ORDSYS.ORDAnnotation;
  n   INTEGER;
BEGIN
 SELECT T.a INTO mya FROM TANNOT T WHERE N = 1;
 n := mya.getAnnotationCount;
 FOR i in 1..n LOOP
   obj := mya.getAnnotationObjByPosition(i);
```

```
   DBMS_OUTPUT.PUT_LINE('Key '||TO_CHAR(i)||':'||obj.key);
   DBMS_OUTPUT.PUT_LINE('Value: '||obj.value);
   DBMS_OUTPUT.PUT_LINE('---------------------------------------');
 END LOOP;
END;
```

getAnnotationKeyByPosition( ) Method

Format

```
getAnnotationKeyByPosition(pos IN INTEGER) RETURN VARCHAR2;
```

Description

Returns the value of the key field by the ORDAnnotation object's position in the ORDAnnotationList table.

Parameters

pos
The ORDAnnotation object's position in the ORDAnnotationList table.

Usage

none

Pragmas

Pragma RESTRICT_REFERENCES(getAnnotationKeyByPosition, WNDS, WNPS, RNPS)

Exception

If the value of pos is greater than the number of ORDAnnotation objects in the ORDAnnotationList table, then calling this method raises a POSITION_OUT_OF_RANGE exception.

Example

Return the value of the key field by the ORDAnnotation object's position in the ORDAnnotationList table:

```
DECLARE
  mya ORDSYS.ORDAnnotations;
  n   INTEGER;
BEGIN
 SELECT T.a INTO mya FROM TANNOT T WHERE N = 1;
 n := mya.getAnnotationCount;
 FOR i in 1..n LOOP
```

```
   DBMS_OUTPUT.PUT_LINE('Key '||TO_CHAR(i)||':'||mya.getAnnotationKeyByPosition(
i));
   DBMS_OUTPUT.PUT_LINE('Value: '||mya.getAnnotationValueByPosition(i));
   DBMS_OUTPUT.PUT_LINE('---------------------------------------');
 END LOOP;
END;
```

getAnnotationValueByPosition( ) Method

Format

```
getAnnotationValueByPosition(pos IN INTEGER) RETURN VARCHAR2;
```

Description

Returns the value of the value field by the ORDAnnotation object's position in the ORDAnnotationList table.

Parameters

pos
The ORDAnnotation object's position in the ORDAnnotationList table.

Usage

none

Pragmas

Pragma RESTRICT_REFERENCES(getAnnotationValueByPosition, WNDS, WNPS, RNPS)

Exception

If the value of pos is greater than the number of ORDAnnotation objects in the ORDAnnotationList table, then calling this method raises a POSITION_OUT_OF_RANGE exception.

Example

Return the value of the value field by the ORDAnnotation object's position in the ORDAnnotationList table:

```
DECLARE
  mya ORDSYS.ORDAnnotations;
  n   INTEGER;
BEGIN
 SELECT T.a INTO mya FROM TANNOT T WHERE N = 1;
 n := mya.getAnnotationCount;
 FOR i in 1..n LOOP
```

```
   DBMS_OUTPUT.PUT_LINE('Key '||TO_CHAR(i)||':'||mya.getAnnotationKeyByPosition(
i));
   DBMS_OUTPUT.PUT_LINE('Value: '||mya.getAnnotationValueByPosition(i));
   DBMS_OUTPUT.PUT_LINE('---------------------------------------');
 END LOOP;
END;
```

G

Frequently Asked Questions

A text file containing a list of frequently asked questions is available on line after installing Oracle8*i* *inter*Media.

This text file can be found as follows:

$ORACLE_HOME/ord/img/admin/imfaq.txt

H

Exceptions and Error Messages

H.1 Exceptions

The following sections describe the exceptions and error messages of *inter*Media objects.

H.1.1 ORDAudioExceptions Exceptions

The following exceptions are associated with the ORDAudio object:

UNSUPPORTED_DATA_SOURCE

Cause: This exception is raised if the data source is external.

Action: Set the source information to a local source.

DESCRIPTION_IS_NOT_SET

Cause: This exception is raised when calling the getDescription function and the description attribute is not set.

Action: Set the description attribute.

INVALID_MIME_TYPE

Cause: This exception is raised if the MIME parameter value of the setMime-Type procedure is NULL.

Action: Set the MIME parameter value to a known value.

AUDIO_FORMAT_IS_NULL

Cause: This exception is raised when calling the getFormat function and the format is NULL.

Action: Set the format for the audio object to a known format.

NULL_INPUT_VALUE

Cause: This exception is raised if the knownFormat parameter value of the setFormat procedure is NULL.

Action: Set these parameters with known values.

METHOD_NOT_SUPPORTED

Cause: This exception is raised when the method called is not supported.

Action: Call a supported method.

AUDIO_PLUGIN_EXCEPTION

Cause: This exception is raised when the audio plug-in raises an exception.

Action: Refer to Section 3.3.1 for more information.

H.1.2 ORDImageExceptions Exceptions

The following exceptions are associated with the ORDImage object:

NULL_LOCAL_DATA

Cause: This exception is raised when source.localData is NULL.

Action: Initialize source.localData with an empty_blob( ).

NULL_PROPERTIES_DESCRIPTION

Cause: This exception is raised when the description parameter to setProperties is not set.

Action: Set the description attribute if you are using a foreign image. Otherwise, do not pass the description parameter.

NULL_DESTINATION

Cause: This exception is raised when the destination image is NULL.

Action: Pass an initialized destination image.

DATA_NOT_LOCAL

Cause: This exception is raised when the source information is not set to local.

Action: Reset the source attribute information to a local image source. Call the import( ) or importFrom( ) method to import the data into the local BLOB.

NULL_CONTENT

Cause: This exception is raised when the content attribute of an ORDImgB or ORDImgF image is NULL.

Action: Initialize the content attribute.

NULL_SOURCE

Cause: This exception is raised when the source image is NULL.

Action: Pass an initialized source image.

H.1.3 ORDVideoExceptions Exceptions

The following exceptions are associated with the ORDVideo object:

UNSUPPORTED_DATA_SOURCE

Cause: This exception is raised if the data source is external.

Action: Set the source information to a local source.

DESCRIPTION_IS_NOT_SET

Cause: This exception is raised when calling the getDescription function and the description attribute is not set.

Action: Set the description attribute.

INVALID_MIME_TYPE

Cause: This exception is raised if the MIME parameter value of the setMime-Type procedure is NULL.

Action: Set the MIME parameter value to a known value.

VIDEO_FORMAT_IS_NULL

Cause: This exception is raised when calling the getFormat function and the format is NULL.

Action: Set the format for the video object to a known format.

NULL_INPUT_VALUE

Cause: This exception is raised if either the knownWidth or knownHeight parameter values of the setFrameSize procedure is NULL.

Action: Set these parameters with known values.

METHOD_NOT_SUPPORTED

Cause: This exception is raised when the method called is not supported.

Action: Call a supported method.

VIDEO_PLUGIN_EXCEPTION

Cause: This exception is raised when the video plug-in raises an exception.

Action: Refer to Section 5.3.1 for more information.

H.1.4 ORDSourceExceptions Exceptions

The following exceptions are associated with the ORDSource object:

INCOMPLETE_SOURCE_INFORMATION

Cause: This exception is raised when the source information (srcType is NULL) is incomplete or srcType is NULL and data is not stored locally in the BLOB.

Action: Check your source information and set srcType, srcLocation, or srcName attributes as needed.

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.

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.

EMPTY_SOURCE

Cause: This exception is raised when the source is local but the source is NULL. This exception can be raised from the audio format plug-in.

Action: Pass an initialized source.

NULL_SOURCE

Cause: This exception is raised when the local source is NULL.

Action: Pass an initialized source.

UNSUPPORTED_DATA_SOURCE

Cause: This exception is raised if the data source is external.

Action: Set the source information to a local source.

INVALID_SOURCE_TYPE

Cause: This exception is raised when the getBFile method detects a source type other than 'FILE'.

Action: Ensure that the source type is 'FILE'.

SOURCE_PLUGIN_EXCEPTION

Cause: This exception is raised when the source plug-in raises an exception.

Action: Refer to Section 6.3.1, Section 6.3.2, and Section 6.3.3 for more information.

H.2 ORDAudio Error Messages

AUD-00001 unable to initialize Audio Cartridge environment

Cause: The audio processing external procedure initialization process failed.

Action: Contact Oracle Customer Support.

AUD-00511 string

Cause: An error was found while accessing audio data.

Action: Contact Oracle Customer Support.

AUD-00599 internal error

Cause: An internal error has occurred.

Action: Contact Oracle Customer Support.

AUD-00601 out of memory while copying audio

Cause: Operating system process memory has been exhausted while copying the audio.

Action: See the database administrator or operating system administrator to increase process memory quota.

AUD-00602 unable to access audio data

Cause: An error occurred while reading or writing audio data.

Action: Contact your system administrator.

AUD-00603 unable to access source audio data

Cause: The SOURCE attribute is invalid.

Action: Ensure that the SOURCE attribute of the source is populated with audio data.

AUD-00604 unable to access destination audio data

Cause: The destination SOURCE attribute is invalid.

Action: Ensure that the SOURCE attribute of the destination source is valid

AUD-00606 unable to access audio data

Cause: An attempt was made to access an invalid audio.

Action: Ensure that the SOURCE attribute of the audio is populated with audio data.

AUD-00607 unable to write to destination audio

Cause: The destination audio SOURCE attribute is invalid.

Action: Ensure that the SOURCE attribute of the destination audio is populated with an initialized BLOB locator and that you have sufficient tablespace.

AUD-00702 unable to initialize audio processing environment

Cause: The audio processing external procedure initialization process failed.

Action: Contact Oracle Customer Support.

AUD-00703 unable to read audio data

Cause: There is no audio data in the source.

Action: Refer to the Oracle8*i interM*edia Audio, Image, and Video User's Guide and Reference for information on how to populate audio data.

AUD-00704 unable to read audio data

Cause: There is no audio data in the source.

Action: Refer to the Oracle8*i interM*edia Audio, Image, and Video User's Guide and Reference for information on how to populate audio data.

AUD-00705 unsupported or corrupted input format

Cause: This is an internal error.

Action: Contact Oracle Customer Support.

AUD-00706 unsupported or corrupted output format

Cause: This is an internal error.

Action: Contact Oracle Customer Support.

AUD-00707 unable to access audio data

Cause: An error occurred while reading or writing audio data.

Action: Contact your system administrator.

AUD-00710 unable write to destination audio

Cause: The destination audio is invalid.

Action: Ensure that the destination audio is valid and it has sufficient storage.

AUD-00711 unable to set properties of destination audio

Cause: This is an internal error.

Action: Contact Oracle Customer Support.

AUD-00712 unable to write to destination audio

Cause: The destination audio is invalid.

Action: Ensure that the destination audio is valid and it has sufficient storage.

AUD-00713 unsupported destination audio format

Cause: A request was made to convert an audio to a format that is not supported.

Action: Refer to the Oracle8*i* *inter*Media Audio, Image, and Video User's Guide and Reference for supported formats.

AUD-00714 internal error

Cause: This is an internal error.

Action: Contact Oracle Customer Support.

AUD-00715 Unable to open audio stored in current source

Cause: The audio stored in the source object could not be opened for reading.

Action: Ensure that the access privileges of the audio source are defined properly for the caller.

H.3 ORDImage Error Messages

IMG-00001, "unable to initialize Oracle8i interMedia environment"

Cause: The image processing external procedure initialization process failed.

Action: Contact Oracle Worldwide Customer Support Services.

IMG-00502, "invalid scale value"

Cause: An invalid scale value was found while parsing the parameters for the image process function.

Action: Correct the statement by using a valid scale value. Refer to the Oracle8*i* *inter*Media Audio, Image, and Video User's Guide and Reference documentation for a description of the correct usage and syntax for the image processing command string.

IMG-00505, "missing value in CUT rectangle"

Cause: An incorrect number of values was used to specify a rectangle.

Action: Use exactly four integer values for the lower left and upper right vertices.

IMG-00506, "extra value in CUT rectangle"

Cause: An incorrect number of values were used to specify a rectangle.

Action: Use exactly four integer values for the lower left and upper right vertices.

IMG-00510, application-specific-message

Cause: A syntax error was found while parsing the parameters for the image process function.

Action: Correct the statement by using valid parameter values. Refer to the Oracle8*i interM*edia Audio, Image, and Video User's Guide and Reference documentation for a description of the correct usage and syntax for the image processing command string.

IMG-00511, application-specific-message

Cause: An error was found while accessing image data.

Action: Contact Oracle Worldwide Customer Support Services.

IMG-00512, "multiple incompatible scaling parameters found"

Cause: Multiple incompatible scaling parameters were found in the image process command string. With the exception of XSCALE and YSCALE which can be used together in a process command string, scaling functions are mutually exclusive and cannot be combined.

Action: Remove scaling functions until only one remains (or two if they are XSCALE and YSCALE).

IMG-00513, "missing value in scaling operation"

Cause: An incorrect number of values was used to specify image dimensions. fixedScale and maxScale require exactly two integer values for the X and Y dimensions of the desired image.

Action: Use two values for fixedScale and maxScale.

IMG-00514, "extra value in scaling operation"

Cause: An incorrect number of values was used to specify image dimensions. fixedScale and maxScale require exactly two integer values for the X and Y dimensions of the desired image.

Action: Use two values for fixedScale and maxScale.

IMG-00515, "incorrect number of input channels"

Cause: An incorrect number of values was used to specify input channels. InputChannels requires either one or three channel numbers for the gray or red, green, and blue channel assignments.

Action: Use either one or three values to specify the input channels.

IMG-00516, "default channel out of range"

Cause: An incorrect value was used to specify the default channel selection.

Action: Use a channel number that is less than or equal to the number of bands.

IMG-00517, "height or width not present in parameter string"

Cause: Height and/or width were not specified in the setProperties parameter string.

Action: Specify both the height and width.

IMG-00518, "invalid value for height or width"

Cause: Height and width must be positive integers.

Action: Specify both the height and width as positive integers.

IMG-00519, "illegal combination of parameters"

Cause: Other than height, width, dataOffset, and userString, no other parameters may be specified in the setProperties parameter string when CCITTG3 or CCITTG4 is used as the compressionFormat.

Action: Supply only the height and width when compressionFormat is either CCITTG3 or CCITTG4. The dataOffset and userString may optionally be supplied as well.

IMG-00531, "empty or null image processing command"

Cause: An empty or null image processing command was passed to the image process function.

Action: Refer to the Oracle8*i* *inter*Media Audio, Image, and Video User's Guide and Reference for a description of the correct usage and syntax for the image processing command string.

IMG-00599, "internal error"

Cause: An internal error has occurred.

Action: Contact Oracle Worldwide Customer Support Services.

IMG-00601, "out of memory while copying image"

Cause: Operating system process memory has been exhausted while copying the image.

Action: See the database administrator or operating system administrator to increase process memory quota.

IMG-00602, "unable to access image data"

Cause: An error occurred while reading or writing image data.

Action: Contact your system administrator.

IMG-00603, "unable to access source image data"

Cause: The source image SOURCE attribute is invalid.

Action: Ensure that the SOURCE attribute of the source image is populated with image data.

IMG-00604, "unable to access destination image data"

Cause: The destination image SOURCE attribute is invalid.

Action: Ensure that the SOURCE attribute of the destination image is populated with image data.

IMG-00606, "unable to access image data"

Cause: An attempt was made to access an invalid image.

Action: Ensure that the SOURCE attribute of the image is populated with image data.

IMG-00607, "unable to write to destination image"

Cause: The destination image SOURCE attribute is invalid.

Action: Ensure that the SOURCE attribute of the destination image is initialized correctly and that you have sufficient tablespace.

IMG-00609, "unable to read image stored in a BFILE"

Cause: The image stored in a BFILE cannot be opened for reading.

Action: Ensure that the access privileges of the image file and the image file's directory allow read access.

IMG-00701, "unable to set the properties of an empty image"

Cause: There is no data in the image object.

Action: Refer to the Oracle8*i* *inter*Media Audio, Image, and Video User's Guide and Reference for information on how to populate image data into the image object.

IMG-00702, "unable to initialize image processing environment"

Cause: The image processing external procedure initialization process failed.

Action: Contact Oracle Worldwide Customer Support Services.

IMG-00703, "unable to read image data"

Cause: There is no image data in the image object.

Action: Refer to the Oracle8*i* *inter*Media Audio, Image, and Video User's Guide and Reference for information on how to populate image data into the image object.

IMG-00704, "unable to read image data"

Cause: There is no image data in the image object.

Action: Refer to the Oracle8*i* *inter*Media Audio, Image, and Video User's Guide and Reference for information on how to populate image data into the image object.

IMG-00705, "unsupported or corrupted input format"

Cause: This is an internal error.

Action: Contact Oracle Worldwide Customer Support Services.

IMG-00706, "unsupported or corrupted output format"

Cause: This is an internal error.

Action: Contact Oracle Worldwide Customer Support Services.

IMG-00707, "unable to access image data"

Cause: An error occurred while reading or writing image data.

Action: Contact your system administrator.

IMG-00710, "unable write to destination image"

Cause: The destination image is invalid.

Action: Ensure that the SOURCE attribute of the destination image is initialized and that you have sufficient tablespace.

IMG-00711, "unable to set properties of destination image"

Cause: This is an internal error.

Action: Contact Oracle Worldwide Customer Support Services.

IMG-00712, "unable to write to destination image"

Cause: The destination image is invalid.

Action: Ensure that the SOURCE attribute of the destination image is initialized and that you have sufficient tablespace.

IMG-00713, "unsupported destination image format"

Cause: A request was made to convert an image to a format that is not supported.

Action: Refer to the Oracle8*i interM*edia Audio, Image, and Video User's Guide and Reference for supported formats.

IMG-00714, "internal error"

Cause: This is an internal error.

Action: Contact Oracle Worldwide Customer Support Services.

IMG-00715, "Unable to open image stored in a BFILE"

Cause: The image stored in a BFILE could not be opened for reading.

Action: Ensure that the access privileges of the image file and the image file's directory allow read access.

IMG-00716, "source image format does not support process options"

Cause: A request was made to apply a processing option not supported by the source image format.

Action: Refer to the Oracle8*i interM*edia Audio, Image, and Video User's Guide and Reference for a discussion of supported processing options.

IMG-00717, "destination image format does not support process options"

Cause: A request was made to apply a processing option not supported by the destination image format.

Action: Refer to the Oracle8*i interM*edia Audio, Image, and Video User's Guide and Reference for a discussion of supported processing options.

IMG-00718, "the same Temporary LOB cannot be used as both source and destination"

Cause: A call was made to processCopy with the same Temporary LOB being specified as both the source and destination.

Action: Specify a different LOB for parameter "dest".

H.4 ORDVideo Error Messages

VID-00001 unable to initialize Video Cartridge environment

Cause: The video processing external procedure initialization process failed.

Action: Contact Oracle Customer Support.

VID-00511 string

Cause: An error was found while accessing video data.

Action: Contact Oracle Customer Support.

VID-00599 internal error

Cause: An internal error has occurred.

Action: Contact Oracle Customer Support.

VID-00601 out of memory while copying video

Cause: Operating system process memory has been exhausted while copying the video.

Action: See the database administrator or operating system administrator to increase process memory quota.

VID-00602 unable to access video data

Cause: An error occurred while reading or writing video data.

Action: Contact your system administrator.

VID-00603 unable to access source video data

Cause: The SOURCE attribute is invalid.

Action: Ensure that the SOURCE attribute of the source is populated with video data.

VID-00604 unable to access destination video data

Cause: The destination SOURCE attribute is invalid.

Action: Ensure that the SOURCE attribute of the destination source is valid.

VID-00606 unable to access image data

Cause: An attempt was made to access an invalid image.

Action: Ensure that the SOURCE attribute of the image is populated with image data.

VID-00607 unable to write to destination image

Cause: The destination image SOURCE attribute is invalid.

Action: Ensure that the SOURCE attribute of the destination image is populated with an initialized BLOB locator and that you have sufficient tablespace.

VID-00702 unable to initialize video processing environment

Cause: The video processing external procedure initialization process failed.

Action: Contact Oracle Customer Support.

VID-00703 unable to read video data

Cause: There is no video data in the source.

Action: Refer to the Oracle8*i interM*edia Audio, Image, and Video User's Guide and Reference for information on how to populate video data.

VID-00704 unable to read video data

Cause: There is no video data in the source.

Action: Refer to the Oracle8*i interM*edia Audio, Image, and Video User's Guide and Reference information on how to populate video data.

VID-00705 unsupported or corrupted input format

Cause: This is an internal error.

Action: Contact Oracle Customer Support.

VID-00706 unsupported or corrupted output format

Cause: This is an internal error.

Action: Contact Oracle Customer Support.

VID-00707 unable to access video data

Cause: An error occurred while reading or writing video data.

Action: Contact your system administrator.

VID-00710 unable write to destination video

Cause: The destination video is invalid.

Action: Ensure that the destination video is valid and it has sufficient storage.

VID-00711 unable to set properties of destination video

Cause: This is an internal error.

Action: Contact Oracle Customer Support.

VID-00712 unable to write to destination video

Cause: The destination video is invalid.

Action: Ensure that the destination video is valid and it has sufficient storage.

VID-00713 unsupported destination video format

Cause: A request was made to convert an video to a format that is not supported.

Action: Refer to the Oracle8*i* *inter*Media Audio, Image, and Video User's Guide and Reference for supported formats.

VID-00714 internal error

Cause: This is an internal error.

Action: Contact Oracle Customer Support.

VID-00715 Unable to open video stored in current source

Cause: The image stored in the source object could not be opened for reading.

Action: Ensure that the access privileges of the video source are defined properly for the caller.

I

Deprecated Image Object Types and Methods

For release 8.1.4 and earlier, Oracle8 Image Cartridge described a set of image object types and methods that are deprecated for release 8.1.5. A deprecated feature is a feature that ships on the software kit and is working for the current release; however, it will not be enhanced in a future release, and may become obsolete and deleted in the future. These deprecated features are described here for reference and to help you migrate your release 8.1.4 and earlier image applications to release 8.1.5 *inter*Media image applications. This appendix describes these deprecated image object types and methods.

These deprecated features consists of two object types:

- ORDImgB supports images stored in an Oracle8 binary large object (BLOB)
- ORDImgF supports images stored in an Oracle8 external binary file (BFILE)

The cartridge (release 8.1.4) includes the following functions and procedures:

Table I–1 Functions and Procedures

Function or Procedure	Description
checkProperties	Verifies the stored image attributes match the actual image.
copyContent	Creates a copy of an image in another BLOB (available only for BLOBs, and not BFILEs).
deleteContent	Deletes the image content.
getMimeType	Returns the MIME type of an image.
getCompressionFormat	Returns the type of compression used on the image.
getContent	Returns the BLOB or BFILE containing the image.

Table I–1 Functions and Procedures

Function or Procedure	Description
getContentFormat	Returns the format of the image.
getContentLength	Returns the size of the image in bytes.
getFileFormat	Returns the file type of an image.
getHeight	Returns the height of the image in pixels.
getWidth	Returns the width of the image in pixels.
process	Performs in-place image processing on a BLOB.
processCopy	Performs image processing while copying an image to another BLOB.
setProperties	Fills in the attribute fields of an image (ORDImgB or ORDImgF data type).

When you are storing or copying images in an ORDImgB object, you must first create an empty BLOB in the table. The examples in this chapter assume that the following table, ordimgtab, has been created to store three images. Three empty rows have been created as follows:

```
create table ordimgtab(col1 number, col2 ORDSYS.ORDImgB);
insert into ordimgtab values
  (1, ORDSYS.ORDImgB(empty_blob(), NULL, NULL, NULL, NULL, NULL, NULL));
insert into ordimgtab values
  (2, ORDSYS.ORDImgB(empty_blob(), NULL, NULL, NULL, NULL, NULL, NULL));
insert into ordimgtab values
  (3, ORDSYS.ORDImgB(empty_blob(), NULL, NULL, NULL, NULL, NULL, NULL));
commit;
```

When storing images in an ORDImgF object, you must populate the type with an initializer.

```
create table ordimgtab(col1 number,col2 ORDSYS.ORDImgF);
insert into ordimgtab values
  (1, ORDSYS.ORDImgF(bfilename
       ('ORDIMGDIR','jdoe.gif'),NULL,NULL,
        NULL,NULL,NULL,NULL));
```

The 'bfilename' argument 'ORDIMGDIR' is a directory referring to a file system directory. Note that the directory name in a bfilename constructor must be in uppercase. The following sequence creates a directory named ORDIMGDIR:

```
connect internal
create or replace directory ORDIMGDIR as '<myimage directory>';
grant read on directory ORDIMGDIR to <user-or-role> with grant option;
```

ORDImgB Object Type

The ORDImgB object type is used for basic storage and retrieval of image data within an Oracle database. This object type is defined as follows:

```
CREATE TYPE ORDImgB AS OBJECT
(
-- TYPE ATTRIBUTES
content             BLOB,
height              INTEGER,
width               INTEGER,
contentLength       INTEGER,
fileFormat          VARCHAR2(64),
contentFormat       VARCHAR2(64),
compressionFormat   VARCHAR2(64),
--- METHOD DECLARATION
  MEMBER PROCEDURE copyContent(dest IN OUT NOCOPY BLOB),
  MEMBER PROCEDURE setProperties(SELF IN OUT ORDImgB),
  MEMBER PROCEDURE process    (SELF    IN OUT ORDImgB,
                               command  IN     VARCHAR2)
  MEMBER PROCEDURE processCopy(command  IN     VARCHAR2,
                               dest     IN OUT NOCOPY BLOB)
  MEMBER FUNCTION  getMimeType RETURN VARCHAR2,
  MEMBER FUNCTION  getContent  RETURN BLOB,
  MEMBER FUNCTION  getContentLength RETURN INTEGER,
  MEMBER PROCEDURE deleteContent (SELF IN OUT ORDImgB),
  MEMBER FUNCTION  getHeight   RETURN INTEGER,
  MEMBER FUNCTION  getWidth    RETURN INTEGER,
  MEMBER FUNCTION  getFileFormat RETURN VARCHAR2,
  MEMBER FUNCTION  getContentFormat RETURN VARCHAR2,
  MEMBER FUNCTION  getCompressionFormat RETURN VARCHAR2,
  MEMBER FUNCTION  checkProperties RETURN BOOLEAN
);
```

where:

- content: is the stored image
- height: is the height of the image in pixels
- width: is the width of image in pixels
- contentLength: is the size of the *on-disk* image file in bytes
- fileFormat: is the file type of image (such as, TIFF, JFIF)

- contentFormat: is the type of image (such as, monochrome, 8-bit grayscale)
- compressionFormat: is the compression type of image

In PL/SQL, data is moved with the DBMS LOB package. From the client, data is moved using OCI LOB calls. The ORDImgB object type does not supply piece-wise routines for moving data.

ORDImgF Object Type

The ORDImgF object type is used for retrieval of image data stored in external files. BFILE images are assumed to be read-only and this is reflected in the member procedures defined on the object type.

```
CREATE TYPE ORDImgF AS OBJECT
(
-- TYPE ATTRIBUTES
content             BFILE,
height              INTEGER,
width               INTEGER,
contentLength       INTEGER,
fileFormat          VARCHAR2(64),
contentFormat       VARCHAR2(64),
compressionFormat   VARCHAR2(64),

-- METHOD DECLARATION
MEMBER PROCEDURE copyContent(dest IN OUT NOCOPY BLOB),
MEMBER PROCEDURE setProperties(SELF IN OUT ORDImgF),
MEMBER PROCEDURE processCopy(command IN VARCHAR2,
                             dest    IN OUT NOCOPY BLOB),
MEMBER FUNCTION  getMimeType RETURN VARCHAR2,
MEMBER FUNCTION  getContent  RETURN BFILE,
MEMBER FUNCTION  getContentLength RETURN INTEGER,
MEMBER FUNCTION  getHeight   RETURN INTEGER,
MEMBER FUNCTION  getWidth    RETURN INTEGER,
MEMBER FUNCTION  getFileFormat RETURN VARCHAR2,
MEMBER FUNCTION  getContentFormat RETURN VARCHAR2,
MEMBER FUNCTION  getCompressionFormat RETURN VARCHAR2,
MEMBER FUNCTION  checkProperties RETURN BOOLEAN

);
```

where:

- content: is the stored image
- height: is the height of the image in pixels
- width: is the width of image in pixels
- contentLength: is the size of the *on-disk* image file in bytes

- fileFormat: is the file type of image (such as, TIFF, JFIF)
- contentFormat: is the type of image (such as, monochrome, 8-bit grayscale)
- compressionFormat: is the compression type of image

checkProperties Method

Format

```
checkProperties RETURN BOOLEAN;
```

Description

Verifies that the properties stored in attributes of the image object match the properties of the image stored in the BLOB or BFILE. This method should not be used for foreign images.

Parameters

none

Returns

BOOLEAN

Usage

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

Example

Check the image attributes.

```
imgb1            ORDSYS.ORDImgB;
properties_match BOOLEAN;

...
properties_match := imgb1.checkProperties;
```

copyContent Method

Format

```
copyContent (dest IN OUT NOCOPY BLOB);
```

Description

Copies an image without changing it.

Parameters

dest
The destination of the new image.

Usage

This method copies the image data into the supplied BLOB.

Example

Create a copy of the image in type image1 into a BLOB called myblob:

```
image1.copyContent(myblob);
```

deleteContent Method

Format

```
deleteContent;
```

Description

Deletes the contents of the image.

Parameters

none

Usage

Use this method to delete the contents of the image BLOB. This method works only with BLOBS, not BFILES.

Example

Delete the image.

```
imgb1  ORDSYS.ORDImgB;

...
imgb1.deleteContent;
```

getCompressionFormat Method

Format

```
getCompressionFormat RETURN VARCHAR2;
```

Description

Returns the compression type of an image. This method does not actually read the LOB, it is a simple accessor method that returns the value of the compressionFormat attribute.

Parameters

none

Returns

VARCHAR2

Usage

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

Example

Get the compression type of an image:

```
imgb1             ORDSYS.ORDImgB;
compressionFormat VARCHAR2(64);

...
compressionFormat := imgb1.getCompressionFormat;
```

getContent Method

Format

```
getContent RETURN BLOB;
getContent RETURN BFILE;
```

Description

Returns the LOB locator of the BLOB or BFILE containing the image. This is a simple accessor method that returns the value of the content attribute.

Parameters

none

Returns

BLOB or BFILE, corresponding to how the image is stored.

Usage

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

Example

Get the LOB locator for an image:

```
imgb1   ORDSYS.ORDImgB;
content BLOB;
...
content := imgb1.getContent;
```

getContentFormat Method

Format

```
getContentFormat RETURN VARCHAR2;
```

Description

Returns the type of an image (such as monochrome or 8-bit grayscale). This method does not actually read the LOB, it is a simple accessor method that returns the value of the contentFormat attribute.

Parameters

none

Returns

VARCHAR2

Usage

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

Example

Get the type of an image:

```
imgb1         ORDSYS.ORDImgB;
contentFormat VARCHAR2(64);

...
contentFormat := imgb1.getContentFormat;
```

getContentLength Method

Format

```
getContentLength RETURN INTEGER;
```

Description

Returns the size of the on-disk image in bytes. This method does not actually read the LOB, it is a simple accessor method that returns the value of the contentLength attribute.

Parameters

none

Returns

INTEGER

Usage

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

Example

Get the content length of an image:

```
imgb1         ORDSYS.ORDImgB;
contentLength INTEGER;

...
contentLength := imgb1.getContentLength;
```

getFileFormat Method

Format

```
getFileFormat RETURN VARCHAR2
```

Description

Returns the file type of an image (such as TIFF or JFIF). This method does not actually read the LOB, it is a simple accessor method that returns the value of the fileFormat attribute.

Parameters

none

Returns

VARCHAR2

Usage

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

Example

Get the file type of an image:

```
imgb1      ORDSYS.ORDImgB;
fileFormat VARCHAR2(64);

...
fileFormat := imgb1.getFileFormat;
```

getHeight Method

Format

```
getHeight RETURN INTEGER;
```

Description

Returns the height of an image in pixels. This method does not actually read the LOB, it is a simple accessor method that returns the value of the height attribute.

Parameters

none

Returns

INTEGER

Usage

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

Example

Get the height of an image:

```
imgb1  ORDSYS.ORDImgB;
height INTEGER;

...
height := imgb1.getHeight;
```

getMimeType Method

Format

```
getMimeType RETURN VARCHAR2;
```

Description

Returns the MIME (Multipurpose Internet Mail Extension) type of an image (such as image/jpeg or image/tiff). This method returns the MIME type based on the file-Format of the image. See Appendix A for the MIME type associated with each supported file format.

Parameters

none

Returns

VARCHAR2

Usage

Use this method to obtain the MIME type of the image. The MIME type is required by Web browsers along with the image content. It tells the Web browser how to interpret the image content.

For unrecognized file formats, this method returns image/binary.

Example

Get the MIME type of an image:

```
imgb1      ORDSYS.ORDImgB;
mimeType   VARCHAR2(64);

...
mimeType := imgb1.getMimeType;
```

getWidth Method

Format

```
getWidth RETURN INTEGER;
```

Description

Returns the width of an image in pixels. This method does not actually read the LOB, it is a simple accessor method that returns the value of the width attribute.

Parameters

none

Returns

INTEGER

Usage

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

Example

Get the width of an image:

```
imgb1  ORDSYS.ORDImgB;
width  INTEGER;

...
width := imgb1.getWidth;
```

process Method

Format

```
process (command IN VARCHAR2 );
```

Description

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

Parameters

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

Usage

You can change one or more of the image attributes shown inTable I–2. Table I–3 shows additional changes that can be made only to raw pixel and foreign images. See Appendix A for information on all the supported format combinations. See Appendix C for a more complete description of each operator.

Table I–2 Image Processing Operators

Operator Name	Usage	Values
compressionFormat	compression type / format	JPEG, SUNRLE, BMPRLE, TARGARLE, LZW, LZWHDIFF, FAX3, FAX4, HUFFMAN3, Packbits, GIFLZW
compressionQuality	compression quality	MAXCOMPRATIO, MAXINTEGRITY, LOWCOMP, MEDCOMP, HIGHCOMP
contentFormat	image type / pixel / data format	MONOCHROME, 8 BITGRAYSCALE, 8 BITGREYSCALE, 8BITLUT, 24BITRGB,
cut	window to cut or crop (origin.x origin.y width height)	(Integer Integer Integer Integer) maximum value is 65535
fileFormat	file format of the image	BMPF, CALS, GIFF, JFIF, PICT, RASF, RPIX, TGAF, TIFF
fixedScale	scale to a specific size in pixels (width, height)	(INTEGER INTEGER)

Table I–2 Image Processing Operators

Operator Name	Usage	Values
maxScale	scale to a specific size in pixels, while maintaining the aspect ratio (maxWidth, maxHeight)	(INTEGER INTEGER)
scale	scale factor (for example, 0.5 or 2.0)	<FLOAT> positive
xScale	X-axis scale factor (default is 1)	<FLOAT> positive
yScale	Y-axis scale factor (default is 1)	<FLOAT> positive

Table I–3 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.	RGB (default), RBG, GRB, GBR, BRG, BGR
InputChannels	For multiband images, specify either one (grayscale) or three integers indicating which channels to assign to red (first), green (second), and blue (third). Note that this parameter affects the source image, not the destination.	INTEGER or INTEGER INTEGER INTEGER
Interleave	Controls band layout within the image: Band Interleaved by Pixel Band Interleaved by Line Band Sequential	BIP (default), BIL, BSQ
PixelOrder	If NORMAL, then the leftmost pixel appears first in the image.	NORMAL (default), REVERSE
ScanlineOrder	If NORMAL, then the top scanline appears first in the image.	NORMAL (default), INVERSE

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

Examples

Change the file format of image1 to GIF:

```
image1.process('fileFormat=GIFF');
```

Change image1 to use lower quality JPEG compression and double the length of the image along the X-axis:

```
image1.process('compressionFormat=JPEG, compressionQuality=LOWCOMP,
xScale="2.0"');
image1.setproperties;
```

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

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

```
image1.process(maxScale="32 32");
```

processCopy Method

Format

```
processCopy (command IN VARCHAR2,
             dest IN OUT NOCOPY BLOB);
```

Description

Process an image BLOB or BFILE to another BLOB.

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

See Table I–2, "Image Processing Operators" and Table I–3, "Additional Image Processing Operators for Raw Pixel and Foreign Images".

When using temporary LOBs, you cannot specify the same temporary LOB as both the source and destination.

Example

Copy an image, changing the file format, compression format, and data format in the destination image:

```
create or replace procedure copyit is
  imgB1     ORDSYS.ORDImgB;
  imgB4     ORDSYS.ORDImgB;
  mycommand   VARCHAR2(400);
begin
  select col2 into imgB1 from ordimgtab  where col1 = 1;
  select col2 into imgB4 from ordimgtab  where col1 = 4 for update;
  command:= 'fileFormat=tiff compressionFormat = packbits
  contentFormat = 8bitlut';
  imgB1.processcopy(mycommand,imgB4.content);
  imgB4.setproperties;
```

```
  update ordimgtab set col2 = imgB4 where col1 = 4;
end;
```

setProperties Method

Format

```
setProperties( );
```

Description

Writes the characteristics of an image (BLOB or BFILE) into the appropriate attribute fields.

Parameters

none

Usage

After you have copied, stored, or processed a native format image, call this procedure to set the current characteristics of the new content.

This procedure sets the following information about an image:

- Height in pixels
- Width in pixels
- Data size of the on-disk image in bytes
- File type (TIFF, JFIF, and so forth)
- Image type (monochrome, 8-bit grayscale, and so forth)
- Compression type (JPEG, LZW, and so forth)

Example

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

```
imgB1 ORDSYS.imgB;
.
.
.
select col2 into imgB1 from ordimgtab where col1 = 1 for update;
imgB1.setProperties;
dbms_output.put_line('image width = '|| imgB1.width );
dbms_output.put_line('image height = '|| imgB1.height );
```

```
dbms_output.put_line('image size = '|| imgB1.contentLength );
dbms_output.put_line('image file type = '|| imgB1.fileFormat );
dbms_output.put_line('image type = '|| imgB1.contentFormat );
dbms_output.put_line('image compression = '|| imgB1.compressionFormat );
```

Example output:

```
image width = 360
image height = 490
image size = 59650
image file type = JFIF
image type = 24BITRGB
image compression = JPEG
```

setProperties( ) Method for Foreign Images

Format

```
SetProperties(description IN VARCHAR2);
```

Description

Allows you to write the characteristics of a foreign image (BLOB or BFILE) into the appropriate attribute fields.

Parameters

description
Specifies the image characteristics to set for the foreign image.

Usage

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

You can set the following image characteristics for foreign images, as shown in Table I–4.

Table I–4 Image Characteristics for Headerless Files

Field	Data Type	Description
CompressionFormat	STRING	Value must be CCITTG3, CCITTG4, or NONE (default).
DataOffset	INTEGER	The offset allows the image to have a header that *inter*Media image does not try to interpret. Set the offset to ignore any potential header. The value must be a positive integer less than the LOB length. Default is zero.
DefaultChannelSelection	INTEGER	For multiband images, specify either one (grayscale) or three integers indicating which channels to assign to red (first), green (second), and blue (third).

Table I–4 Image Characteristics for Headerless Files (Cont.)

Field	Data Type	Description
Height	INTEGER	Height of the image in pixels. Value must be a positive integer. There is no default, and a value must be specified.
Interleaving	STRING	Band layout within the image. Valid styles are: ■ BIP (default) Band Interleaved by Pixel ■ BIL Band Interleaved by Line ■ BSQ Band Sequential
NumberOfBands	INTEGER	Value must be a positive integer less than 255 describing the number of color bands in the image. Default is 3.
PixelOrder	STRING	If NORMAL (default), the leftmost pixel appears first in the file. If REVERSE, the rightmost pixel appears first.
ScanlineOrder	STRING	If NORMAL (default), the top scanline appears first in the file. If INVERSE, then the bottom scanline appears first.
UserString	STRING	A 4-character descriptive string. If used, the string is stored in the fileFormat field, appended to the file format ("OTHER:"). Default is blank.
Width	INTEGER	Width of the image in pixels. Value must be a positive integer. There is no default, and a value must be specified.

The values supplied to setProperties( ) are written to the existing ORDImgB and ORDImgF object attributes. The fileFormat is set to "OTHER:" and includes the user string, if supplied.

Example

Select the image type, and then set the attributes using the setProperties method.

```
imgB1 ORDSYS.ORDIMgB;
select col2 into imgB1 from ordimgtab where col1 = 1 for update;
imgB1.setProperties('width=380 height=407 dataOffset=128 bandOrder=BIL
     userString="LSAT"');
```

Glossary

Some glossary entries are from the free online Dictionary of Computing located at: http://wombat.doc.ic.ac.uk/foldoc/index.html.

AAF

Advanced Authoring Format, a professional multimedia authoring format defined by Microsoft Corporation for the Windows operating system to succeed the AVI file format. AAF is an open, industry-developed format that enables the exchange of media among digital production tools and content creation applications.

A-Law

The European and worldwide (not North America and Japan) standard for nonuniform quantising logarithmic compression. A-Law is used as the European telephony standard.

AC-3 (MPEG-2 Audio)/MP3

Surround sound audio.

ADPCM

Adaptive Differential Pulse Code Modulation. A compression technique that records only the difference between samples and adjusts the coding scale dynamically to accommodate large and small differences. ADPCM is simple to implement, but introduces much white noise.

ADT

Abstract data type. *See* also (object type).

AIFF

Audio Interchange File Format. A file format developed by Apple Computer Inc. for storing high quality sampled audio and musical instrument information.

AIFF-C

Audio Interchange File Format with Compression. A file format developed by Apple Computer, Inc., for storing high quality sampled audio and musical instrument information that uses compression.

ASF

Advanced Streaming Format. A multimedia streaming format defined by Microsoft Corporation for the Windows operating system to succeed the AVI file format. ASF is an open, industry-developed format specifically tuned for streaming media distribution.

aspect ratio

Proportions of height versus width.

attributes

Components of object types that are built-in data types or other user-defined types that model the structure of the real world. See *Oracle8i Concepts* for more information.

AU

Audio file format common to the UNIX platform, especially Sun and NeXT computing platforms.

audio

Sound. Computers (and audio compact discs and digital audio tape) handle sound by storing a sequence of discrete samples. The continuous sound waveform from the original source is sampled tens of thousands of times a second. Each sample represents the intensity of the sound pressure wave at that instant. Apart from the sampling frequency, the other parameter is the digital encoding of each sample including the number of bits used. The encoding may be linear, logarithmic or mu-law. The source could be a piece of music, voice recording, or generated audio. *See also* mu-law or u-law.

audio processing

A change to the properties of audio data, such as format, encoding type, number of channels, sampling rate, or sample size.

AVI

Audio Video Interleaved. A file container format defined by Microsoft Corporation. An AVI file typically includes both an audio and a video sequence. The audio and video data can be in several formats, such as Intel Indeo, Iterated Systems ClearVideo (fractal), or Motion JPEG. *See also* AAF and ASF for more information.

bandwidth

The difference between the highest and lowest frequencies of a transmission channel.

BFILE (binary file)

A large object whose value is composed of binary data that is stored outside the database in an operating system file. The file itself is not stored in the database, but a pointer to the file is stored in the database. Because they are outside of the database, BFILEs are read-only.

B-frame

Bidirectionally predictive-coded frame. In MPEG video, a B-frame is encoded using data describing how its picture has changed from the picture in the closest two I-frames or P-frames, one in the past and one in the future.

binary large object

See BLOB.

BLOB

Binary large object. Large objects are stored in the database tablespace in a way that optimizes space and provides efficient access. Only a pointer to the object is actually stored in the row.

CLOB

A large object whose value is composed of single-byte fixed-width character data that corresponds to the database character set defined for the Oracle8 database.

codec

COmpressor DECompressor. Software modules that perform compression and decompression of multimedia data. Some common audio codecs or encoding types are ADPCM, Mu-Law, Two's Compliment, AC-3, and others. Some common video codecs or encoding types are MPEG-1, MPEG-2, Intel Indeo, Iterated Systems ClearVideo (fractal), Radius Cinepak, Motion JPEG, and others. Some common image codecs are JPEG, Packbits, HUFFMAN3, and others.

compression

The coding of data to save storage space or transmission time. Compressed data must be decompressed before it can be used.

compression format

The format of an encoded audio, image, or video file produced by a codec. The Oracle Video Server, for example, supports several compression formats including MPEG-1, MPEG-2, Intel Indeo, Iterated Systems ClearVideo (fractal), Radius Cinepak, and Motion JPEG.

container format

The way in which an encoder multiplexes video and audio together. For example, the container formats supported by the Oracle Video Server include MPEG-1 system, MPEG-2 transport, Oracle Video Stream, and any container format that meets the Raw Key Frame requirements.

content format

A description of the image data, such as the pixel or color format.

cropping

Selecting the portion of an image within a specified rectangle and removing everything outside of that rectangle. *See also* cutting.

cutting

Selecting the portion of an image within a specified rectangle to create a subimage. If the subimage is copied to a new image, the effect is a cut. If the subimage replaces the original, the effect is a crop.

default constructor

A method that creates an empty instance of an object.

duration

The time it takes to play the entire audio or video data object.

encoding type

The particular codec algorithm used to encode the audio and video data. Some common audio codecs are ADPCM, Mu-Law, Two's Compliment, AC-3, and others. Some common video codecs are Indeo, MPEG, and others. For audio and video data, usually the encoding type is specific to the computer platform.

export

To move data from the Oracle database to external storage as files, such as to a local file system, or to an HTTP server.

file format

The file format of the audio data, such as AIFF, AIFF-C, AU, WAV, MIDI, RealAudio, and SND. The file format of the video data, such as QuickTime, AVI, MPEG1, and Real Video. The file format of the image data such as (TIFF, JFIF). See Appendix A for supported audio file and compression formats. See Appendix B for supported image file and compression formats.

format

A specific data structure or organization; usually an industry standard such as AIFF-C for audio data or QuickTime, AVI, or MPEG1 for video data.

frame rate

The number of frames displayed every second. Thirty frames / second is considered a good rate for the human eye. A lower frame rate requires less bandwidth but reduces the quality of the video data. A higher frame rate produces a smoother, higher quality video, but requires more processing power and a greater bandwidth.

frame resolution

The number of pixels per inch of frames in the video data. Frame resolution can be adjusted to fit bandwidth requirements. Reducing frame resolution reduces video quality.

frame size

The height and width in pixels of each frame in the video data. This information is useful in estimating bandwidth requirements and determining appropriate display size for a given display screen size.

HTTP header

A body of information that a browser sends along with a URL when requesting a Web page. It includes such information as the browser type and MIME types.

I-frame

Intra-coded frames. In MPEG video, an I-frame is a frame coded as a still image, not using any past history.

I-frame regularity

To ensure good rate control performance, I-frames must occur at regular intervals throughout a video stream. For the US and Japan, there are usually 11 frames (B-frames and P-frames) between each pair of I-frames in a sequence such as: IBBPBBPBBPBBIBBPBBP.... This is based on a random access requirement that an I-frame (starting point) is needed at least once every 0.4 seconds or so. MPEG encoding software often enables you to configure the I-frame frequency.

IDL

Interface Definition Language.

image

A graphic picture. The source could be a photograph, drawing, or generated image.

image processing

Changing the properties of an image, such as through scaling, rotation, or compression.

import

To move data into an Oracle database from external storage, such as external file servers, media servers, HTTP servers, FTP servers, and so forth.

JPEG

Joint Photographic Experts Group. The original name of the committee that designed the standard image compression algorithm. The filename extension is .jpg or .jpeg. JPEG is a static-image compression format.

LOB

Large object. LOBs are used to hold large amounts of raw, binary data. Oracle8*i* *inter*Media supports LOBs.

localData

A BLOB attribute that can keep up to 4GB of image, audio, or video data stored within the Oracle database. All data within localData is protected by Oracle's security and transaction support.

local file system

A file system that allows local access where, for example, BFILES can be stored.

local source

The location of image, audio, or video data within the Oracle database.

lossless compression

A data compression algorithm that retains all the information in the data, allowing it to be recovered perfectly by decompression.

A means for reducing the storage space required for an image. The decompressed image is bit-for-bit identical to the original.

Unix compress and GNU gzip perform lossless compression.

lossy compression

A data compression algorithm that actually reduces the amount of information in the data, rather than just the number of bits used to represent that information. The lost information is usually removed because it is subjectively less important to the quality of the data (usually an image or sound) or because it can be recovered reasonably by interpolation from the remaining data.

A means for reducing the storage space required for an image. The decompressed image is not bit-for-bit identical to the original. Some details are lost or changed, although this might not be noticeable to the naked eye.

MPEG and JPEG are examples of lossy compression techniques.

methods

Components of an object type that are functions or procedures written in PL/SQL and stored in the database or written in a language like C and stored externally. Methods implement specific operations that an application can perform on the data. See *Oracle8i Concepts* for more information.

MIME

Multipurpose Internet Mail Extensions. *See* Multipurpose Internet Mail Extensions (MIME).

MIME Type

A file format defined by the Multipurpose Internet Mail Extensions standard. This unique identifier is used for different file types when conveyed across a MIME-based protocol such as MIME e-mail or HTTP.

moving JPEG, motion JPEG, M-JPEG

A compression technique for moving images that applies JPEG still image compression to each frame of a moving picture sequence.

Play-back requires a machine capable of decompressing and displaying each JPEG image quickly enough to sustain the required frame rate of the picture sequence.

There is no standard for Moving JPEG as with JPEG, but there are JPEG compression chips which are designed to work at television frame rates and resolutions.

MPEG

Moving Picture Expert's Group, an ISO body focused on developing compression formats for full-motion video and audio signals and the synchronization of these signals during playback.

MPEG-1

The first version of the MPEG format, optimized for CD-ROM. It uses discrete cosine transform (DCT) and Huffman coding to remove spatially redundant data within a frame and block-based motion compensated prediction (MCP) to remove data which is temporally redundant between frames. Audio is compressed using subband encoding.

MPEG-2

A variant of the MPEG video and audio compression algorithm and file format, optimized for broadcast quality video for digital storage media up to 4.0Mbits/second. The file extension is .MP2.

MPEG-2 has been approved as International Standard IS-13818.

MPEG-3

A variant of the MPEG video and audio compression algorithm and file format. The file extension is .MP3. This variant no longer exists and has been merged into MPEG-2.

MPEG-4

A variant of the MPEG video and audio compression algorithm and file format used for low bandwidth video telephony. The file extension is .MP4. The International Standards Organization (ISO) has adopted the QuickTime 3 file format to form the starting point for a unified digital media storage format for the MPEG-4 specification. *See also* QuickTime 3 for more information.

mu-law or u-law

The North America and Japan standard for nonuniform quantising logarithmic compression.

Multipurpose Internet Mail Extensions (MIME)

An Internet specification describing file types. Servers and browsers read the MIME type in the file header and decide what to do with the file, such as displaying it with a viewer or playing it as an audio file. MIME is used by HTTP servers to describe the type of file being delivered.

number of channels

Number of audio channels present in the formatted audio data; also known as the channel count. Number of channels may range from 1 for mono audio data to 6 for AC3 encoded surround sound audio data.

number of frames

The number of frames in the video object.

object relational database

A database having both object-oriented and relational characteristics. Objects can be defined and stored, and then retrieved using standard relational methods.

object type

A data type that includes attributes of the data and methods (functions and procedures) for operating on the data.

OCI

Oracle Call Interface.

An object type is sometimes referred to as an abstract data type (ADT).

Oracle Video Server

See OVS.

Oracle Web Application Server

An HTTP server that enables you to create and deploy distributed, cross-platform, dynamic applications. It provides a framework that encompasses a modular distributed architecture, an open API that enables you to create portable applications, and different application models or paradigms. Oracle Web Application Server consists of three components: HTTP Daemons (UNIX) or Listeners, Web Request Broker (WRB), and options or server-side applications.

ORDAnnotations

An *inter*Media-defined object type that is part of the demo program that encapsulates all the functionality required to store and manage audio annotation data.

ORDAudio

A *inter*Media audio-defined object type that encapsulates all the functionality required to store and manage audio data.

ORDImage

A *inter*Media image-defined object type that encapsulates all the functionality required to store and manage image data.

ORDVideo

A *inter*Media video-defined object type that encapsulates all the functionality required to store and manage video data.

ORDSource

A *inter*Media-defined object type that only knows about the source of data; it is a common object type for *inter*Media object types. The ORDSource object type is used to get multimedia data from sources such as BLOBs stored locally in the Oracle database, BFILES stored locally in external file systems, media servers, FTP servers, and HTTP servers.

OVS

Oracle Video Server. An Oracle product that provides an end-to-end software solution for networked client and server computers that store, manage, deliver, and display digital video and audio on demand. Client applications run a wide variety of consumer and corporate platforms. OVS is supported on a variety of server platforms and scales to serve many concurrent users.

P-frame

Predictive-coded frames. In MPEG video, a P-frame is encoded using data describing how its picture has changed from the picture in a previous I-frame or P-frame.

QuickTime

A file container format defined by Apple Computer, Inc., for integrating full-motion video and digitized sound into application programs.

QuickTime 3

An industry standard file format defined by Apple Computer, Inc., to create and publish digital media for the Mac OS, Java, and Windows operating environments. The International Standards Organization (ISO) has adopted the QuickTime 3 file format to form the starting point for a unified digital media storage format for the MPEG-4 specification. This will allow the vast majority of existing hardware, software, and digital content to work seamlessly with this next generation version of MPEG. *See also* MPEG-4 for more information.

Raw Key Frame (RKF)

A set of rules to which the content of video container format can conform. Some video servers, such as the Oracle Video Server, can only play content in container formats that conform to the rules of RKF. The rules of RKF are:

- **stateless** - all the data for displaying the picture in a video frame is contained entirely in that frame, rather than in previous frames.
- **contiguous** - all the data for a frame is stored together in the video file and no other data is mixed with it.

Raw Pixel format

An uncompressed image format with a simple, fixed-size header.

sampling rate

The rate in samples per second at which the audio data was recorded. The sampling rate ranges from 5500 (one fourth of the Mac sampling rate) to 48000 (Digital Audio Tape (DAT) sampling rate).

sample size

The number of samples of audio data present in the audio data, or, stated another way, the number of bits per sample. The number of bits per sample is either 8 (8-bit) or 16 (16-bit).

scaling

Changing the proportions of an image in one or both dimensions. To enlarge an image, scale by a factor greater than one. To shrink an image, scale by a factor between zero and one.

SND

Sound. Audio file format in use on the UNIX platform, especially Sun and NeXT computing platforms.

uniform resource identifier (URI)

A compact string representation of a location (URL) for use in identifying an abstract or physical resource. URI is one of many addressing schemes, or protocols, invented for the Internet for the purpose of accessing objects using an encoded address string.

uniform resource locator (URL)

A form of URI. A compact string representation of the location for a resource that is available on the Internet. It is also the text-string format clients use to encode requests to Oracle Application Server.

video

Streaming images. The source could be a video camera, a recording, or a generated animation. *See also* full-motion video.

video compression

Compression of sequences of images.

WAV

Waveform-audio. WAV is a standard file container format developed by Microsoft Corporation for storing digital audio files. Conversion tools are available to allow most other operating systems to play WAV files.

Index

A

B

C

D

E

F

G

I

J

L

M

O

P

Q

R

S

T

W