Oracle® *inter*Media Audio, Image, and Video Java Classes

User's Guide and Reference

Release 8.1.7

September 2000

Part No. A85374-01

Oracle *inter*Media Audio, Image, and Video Java Classes is a component of Oracle *inter*Media, a product designed to manage multimedia Web content within Oracle.

ORACLE®

Oracle *inter*Media Audio, Image, and Video Java Classes User's Guide and Reference, Release 8.1.7

Part No. A85374-01

Primary Author: Max Chittister

Contributors: Raja Chatterje, Sue Mavris, Dan Mullen, Susan Shepard, Brenda Silva, Rod Ward

Contents

2 Program Examples Using Java Classes

3 ORDAudio Reference Information

4 ORDImage Reference Information

5 ORDVideo Reference Information

List of Examples

Send Us Your Comments

Oracle *inter*Media Audio, Image, and Video Java Classes User's Guide and Reference, Release 8.1.7

Part No. A85374-01

Oracle Corporation welcomes your comments and suggestions on the quality and usefulness of this document. 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?

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

- Electronic mail: nedc_doc@us.oracle.com
- FAX: 603.897.3316 Attn: Oracle *inter*Media Documentation
- Postal service:

  Oracle Corporation
  Oracle *inter*Media Documentation
  One Oracle Drive
  Nashua, NH 03062-2698
  USA

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

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

Preface

This guide describes how to use Oracle *inter*Media Audio, Image, and Video Java Classes.

Audience

This guide is for developers or database administrators who are interested in storing, retrieving, and manipulating audio, image, and video data in an Oracle database, including developers of audio, image, or video specialization applications. Users of this guide should have experience with Java and JDBC.

Organization

This guide contains the following chapters and appendixes:

Chapter 1	Contains a general introduction.
Chapter 2	Contains information on the examples included with the Java Classes installation.
Chapter 3	Contains reference information on the OrdAudio class.
Chapter 4	Contains reference information on the OrdImage class.
Chapter 5	Contains reference information on the OrdVideo class.
Appendix A	Contains information on running the sample files included with the Java Classes.
Appendix B	Contains information on possible exceptions and errors.
Appendix C	Contains information on methods that have been deprecated since the previous release.

Related Documents

This guide is not intended as a standalone document. It is a supplement to *Oracle8i interMedia Audio, Image, and Video User's Guide and Reference*. You need both guides to successfully perform operations on *inter*Media objects using the Java interface to this product.

For more information about using these data options in a development environment, see the following documents in the Oracle8*i* documentation set:

- *Oracle Call Interface Programmer's Guide*
- *Oracle8i Application Developer's Guide - Fundamentals*
- *Oracle8i Concepts*
- *PL/SQL User's Guide and Reference*

For more information on using JDBC, see *Oracle8i JDBC Developer's Guide and Reference*.

Conventions

In this guide, Oracle *inter*Media Audio, Image, and Video Java Classes is sometimes referred to as *inter*Media Java Classes. Oracle *inter*Media Audio, Image and Video 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. In code examples, a boldface number in brackets (for example, **[1]**) indicates that particular code will be described in more detail in the subsequent numbered list.
italic text	Italic text is used for emphasis and for book titles.

Convention	Meaning
< >	Angle brackets enclose user-supplied names.
[ ]	Brackets enclose optional clauses from which you can choose one or none.

1

Introduction

Oracle *inter*Media provides Audio, Image, and Video Java Classes (or *inter*Media Java Classes) to enable users to write Java applications using *inter*Media audio, image, and video objects.

1.1 Oracle *inter*Media Audio, Image, and Video

The capabilities of *inter*Media Audio, Image, and Video include the storage, retrieval, management, and manipulation of multimedia data managed by Oracle8*i*. Oracle *inter*Media supports 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 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 Oracle Video Server

*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)

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

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

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 characteristics include format, encoding type, number of channels, sampling rate, sample size, compression type, and audio duration.

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. *inter*Media audio can store and retrieve audio data of any data format. *inter*Media audio can automatically extract metadata from audio data of a variety of popular audio formats. *inter*Media audio can also extract application attributes and store them in the comments field of the object in XML form identical to what is provided by the *inter*Media Annotator utility. Supported audio attributes depend upon available hardware capabilities or processing power for any user-defined formats. See *Oracle8i interMedia Audio, Image, and Video User's Guide and Reference* for a list of supported data formats from which *inter*Media audio can extract and store attributes and other audio features.

*inter*Media audio is extensible and can support additional audio formats.

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 integrates the storage, retrieval, and management of digitized images in Oracle databases using Oracle8*i*.

*inter*Media image supports two-dimensional, static, digitized raster 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 store or retrieve image data of any data format. *inter*Media image can process and automatically extract properties of images of a variety of popular formats. See *Oracle8i interMedia Audio, Image, and Video User's Guide and Reference* for a list of supported data formats for which *inter*Media image can process and extract metadata. In addition, certain foreign images (formats not natively understood by *inter*Media image) have limited support for image processing.

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 use digitized images within these. *inter*Media image supports storage and retrieval of all formats, as well as processing and attribute extraction of many of those formats.

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

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

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 that picked up by a video camera or video recorded on magnetic media, and convert it into digital values with specific video characteristics. Such characteristics include format, encoding type, frame rate, frame size (width and height), frame resolution, video length, compression type, number of colors, and bit rate.

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, numbers of colors, and bit rates, depending upon how the video data was digitally recorded. *inter*Media video can store and retrieve video data of any data format. *inter*Media video can automatically extract metadata from video data of a variety of popular video formats. *inter*Media video can also extract application attributes and store them in the comments field of the object in XML form identical to what is provided by the *inter*Media Annotator utility. Supported video attributes depend upon available hardware capabilities or processing power for any user-defined formats. See *Oracle8i interMedia Audio, Image, and Video User's Guide and Reference* for a list of supported data formats from which *inter*Media video can extract and store attributes and other video features.

*inter*Media video is extensible and can support additional video formats.

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 Java Application Support

Oracle *inter*Media Audio, Image, and Video lets you store your multimedia information in a database table. However, in addition to storing this data, you might want to retrieve it or modify it. *inter*Media Java Classes lets you write your own Java applications to use, manipulate, and modify multimedia data stored in an Oracle database.

*inter*Media Java Classes lets you connect to a database through JDBC calls, select a database *inter*Media object to become a Java application object, perform various operations on the application object, and commit your changes to the database object.

1.6 Interaction Between Database and Java Application

Perform the following operations to make a connection between a database object and a Java application object:

1. Make a connection from the Java application to the Oracle database through JDBC calls.

   Write a method that returns a valid OracleConnection object; see Example 2–3 for an example of a method that makes a connection to the database.

2. Execute a SELECT statement on the database table and store the results in your Java application.

   To execute the SELECT statement, you must create a Statement object in your application, use the executeQuery( ) method to execute the SELECT statement, and put the results into an OracleResultSet object. See steps 1 and 2 of Example 2–4 for an example.

3. Move the results into an *inter*Media object with the getCustomDatum( ) method.

   Create a Java application *inter*Media object, and instantiate it with the results of the getCustomDatum( ) method. See step 5 of Example 2–4 for an example.

   You should now have a Java application *inter*Media object that is identical to the database object.

4. Perform operations on the Java application object.

5. Update the database object to include the results of the operations in step 4.

Create an OraclePreparedStatement object that contains a SQL statement that updates the database object, and execute the statement. See step 9 of Example 2–4 for an example.

6. Commit your changes.

   Unless you set the setAutoCommit( ) method to TRUE in your connection method, you must perform an explicit commit to update the database object with any changes that you made to the application object. You do this with the JDBC commit( ) method. See step 4 of Example 2–2 for an example of the commit( ) method.

7. Close the connection.

   Close the connection between the Java application and the database with the JDBC close( ) method. See step 5 of Example 2–2 for an example of the close( ) method.

   For more information on using JDBC, see *Oracle8i JDBC Developer's Guide and Reference*.

1.7 Compatibility with Previous Releases of *inter*Media

Oracle Corporation may improve the *inter*Media object types by adding new object attributes in a future release of *inter*Media. Client-side applications that want to maintain compatibility with the 8.1.7 release of the *inter*Media object types (OrdAudio, OrdImage, OrdVideo, and OrdSource), even after a server upgrade that changes the object types, should make a call to the compatibility initialization function at the beginning of the application.

Note: If you do not follow the recommended actions, you may have to upgrade and perhaps even recompile your application when you upgrade to a newer server release that enhances the *inter*Media object types.

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

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

This Java function takes an OracleConnection as an argument. The *inter*Media 8.1.7 Java API will ensure compatibility of your 8.1.7 Java application with any future release of *inter*Media, regardless of enhanced object types.

See step 2 of Example 2–2 for an example of the imCompatibilityInit( ) method.

2

Program Examples Using Java Classes

This chapter provides full-length examples of user-defined classes using *inter*Media Java Classes. Sample SQL scripts that demonstrate how to set up a schema on your database server are also included.

This code will not necessarily match the code shipped as AudioExample.java, ImageExample.java, or VideoExample.java with the *inter*Media Java Classes installation. If you want to run an example on your system, use the files provided with the *inter*Media Java Classes installation; do not attempt to compile and run the code presented in this chapter.

Note: This chapter contains examples of Java and SQL code. Some of the code examples display boldface numbers enclosed in brackets; these indicate that further explanation of that code will be in the numbered list immediately following the example.

2.1 Audio Example

The audio example (including AudioExample.sql and AudioExample.java) contains user-defined methods that use SQL, JDBC, and *inter*Media Java Classes APIs to perform the following operations:

- Create a database server table that contains test content
- Load data into both application and database ORDAudio objects from a local file and set the local field on both the application and database objects
- Load data into both application and database ORDAudio objects from a local stream and set the local field on both the application and database objects
- Load data into both application and database ORDAudio objects from a local byte array and set the local field on both the application and database objects

- Extract and print properties from the application ORDAudio object
- Demonstrate error handling through a failed call to a database method

2.1.1 AudioExample.sql

Example 2–1 shows the complete contents of the AudioExample.sql sample file.

Example 2–1 Contents of AudioExample.sql

```
set echo on

-- PLEASE change system password
connect system/manager
drop user  AUDIOUSER cascade;

[1] create user AUDIOUSER identified by AUDIOUSER;
grant connect,resource to AUDIOUSER identified by AUDIOUSER;

[2] connect AUDIOUSER/AUDIOUSER

[3] CREATE TABLE TAUD(n NUMBER, aud ORDSYS.ORDAUDIO);

--
-- Note - the OrdAudio.init method was added in interMedia 8.1.7.
-- If you are running against an older release of interMedia and Oracle,
-- you will have to modify the following INSERT statements to use the
-- ORDAudio default constructor.
--
[4] INSERT INTO TAUD VALUES(1, ORDSYS.ORDAudio.init( ));
INSERT INTO TAUD VALUES(2, ORDSYS.ORDAudio.init( ));
INSERT INTO TAUD VALUES(3, ORDSYS.ORDAudio.init( ));
commit;
```

The SQL statements in AudioExample.sql perform the following operations:

1. Create a user named AUDIOUSER and grant the appropriate permissions to the user.
2. Connect to the database server as AUDIOUSER.
3. Create a table named TAUD with two columns: a column of numbers and a column of ORDAudio objects.

4. Add three rows to the table, each containing an empty ORDAudio object. The names of the variables being set are included in comments for the first INSERT statement only.

The ORDAudio.init method was added in release 8.1.7. If you are running against a previous release of *inter*Media and Oracle8*i*, you will have to modify the INSERT statements in step 4 to use the ORDAudio default constructor.

See *Oracle8i interMedia Audio, Image, and Video User's Guide and Reference* for more information on the init method.

2.1.2 AudioExample.java

Section 2.1.2.1 through Section 2.1.2.8 show the methods contained in the AudioExample.java sample file.

2.1.2.1 main( ) Method

Example 2–2 shows the main( ) method.

Example 2–2 main( ) Method (Audio)

```
public static void main (String args[ ]){
     byte[ ] ctx = new byte[4000];
     OracleConnection con = null;
     try {
          AudioExample tk = new AudioExample( );
          [1] con = tk.connect( );
          //Include the following line only if you are running
          //an Oracle 8.1.7 database or later.
          //If you are running a database server prior to 8.1.7,
          //the call will fail.
          [2] OrdMediaUtil.imCompatibilityInit(con);
          [3] tk.loadDataFromFile(con);
          tk.extractProperties(con);
          tk.printProperties(con);
          tk.otherMethods(con);
          tk.loadDataFromStream(con);
          tk.loadDataFromByteArray(con);
          [4] con.commit( );
          [5] con.close( );
          System.out.println("Done.");
     }
     [6] catch (Exception e) {
          try {
```

```
                System.out.println("Exception : " + e);
                con.close( );
           }
           catch(Exception ex) {
                System.out.println("Close Connection Exception : " + ex);
           }
      }
}
```

The code in the main( ) method performs the following operations:

1. Uses the connect( ) method to make a connection to a database table.
2. Ensures the compatibility of your 8.1.7 application. See Section 1.7 for more information.
3. Calls several methods (also defined in AudioExample.java) that manipulate objects on the database server and the local machine.
4. Commits any changes made to the database table.
5. Closes the connection to the database.
6. Handles any errors or exceptions raised by the code.

Section 2.1.2.2 through Section 2.1.2.8 will provide information on the methods called from the main( ) method in the order in which they are called, not in the order they appear in AudioExample.java.

2.1.2.2 connect( ) Method

Example 2–3 shows a user-defined method named connect( ), which makes a connection from the application to the database.

Example 2–3 connect( ) Method (Audio)

```
public OracleConnection connect( ) throws Exception {
     String connectString;
     [1] Class.forName ("oracle.jdbc.driver.OracleDriver");
     [2] connectString = "jdbc:oracle:oci8:@";
     [3] OracleConnection con = (OracleConnection)DriverManager.getConnection
          (connectString,"AUDIOUSER","AUDIOUSER");
     [4] con.setAutoCommit(false);
     return con;
}
```

The connect( ) method performs the following operations:

1. Loads the JDBC drivers directly, because Oracle uses a JDK-compliant Java virtual machine.
2. Defines a string that contains the URL of the database to which you will connect. You may need to change this string to match your database.
3. Sets the connection to the database, using the URL contained in connectString, the user name AUDIOUSER, and the password AUDIOUSER. The user name and password were created by AudioExample.sql.
4. Disables the auto-commit mode. This means that you must commit or roll back manually with the commit( ) or rollback( ) methods, respectively.

2.1.2.3 loadDataFromFile( ) Method

Example 2–4 shows a user-defined method named loadDataFromFile( ), which uses the *inter*Media loadDataFromFile( ) method to populate the application object with media data.

Example 2–4 loadDataFromFile( ) Method (Audio)

```
public void loadDataFromFile(OracleConnection con) {
     try {
          [1] Statement s = con.createStatement( );
          [2] OracleResultSet rs = (OracleResultSet) s.executeQuery
               ("select * from TAUD where n = 1 for update ");
          int index = 0;
          [3] while(rs.next( )){
               [4] index = rs.getInt(1);
               [5] OrdAudio audObj = (OrdAudio) rs.getCustomDatum
                    (2, OrdAudio.getFactory( ));
               [6] audObj.loadDataFromFile("testaud.dat");
               [7] audObj.getDataInFile("output1.dat");
               System.out.println("************AFTER getDataInFile ");
               [8] System.out.println(" getContentLength output : " +
                    audObj.getContentLength( ));
               [9] OraclePreparedStatement stmt1 = (OraclePreparedStatement)
                    con.prepareCall("update taud set aud = ? where
                    n = " + index);
               stmt1.setCustomDatum(1,audObj);
               stmt1.execute( );
               stmt1.close( ) ;
               index++;
          }
          System.out.println("loading successful");
```

```
      }
      [10] catch(Exception e) {
           System.out.println("exception raised " + e);
           System.out.println("loading unsuccessful");
      }
}
```

The loadDataFromFile( ) method performs the following operations:

1. Creates an OracleStatement object.
2. Executes the given SQL query and puts the results into a local OracleResultSet object.
3. Performs the operations in the loop while there are results in the OracleResultSet that have not been processed. However, in this case, there is only one row included in the OracleResultSet, so the operations in the loop will run once.
4. Sets an index variable to the value of the integer in the first column of the first row in the OracleResultSet (in this case, the value is 1).
5. Creates a local ORDAudio object named audObj. Populates audObj with the contents of the ORDAudio object in the second column of the current row in the OracleResultSet.
6. Uses the ORDAudio loadDataFromFile( ) method to load the media data in testaud.dat into the database ORDAudio object and into audObj. This also sets the local field on audObj, but not the database object.
7. Uses the getDataInFile( ) method to get the media data from audObj and loads it into a file on the local system named output1.dat.
8. Gets the content length of audObj and prints it to the screen to verify the success of the loading.
9. Creates and executes a SQL statement that will update the database ORDAudio object with the contents of audObj.
10. Handles any errors or exceptions raised by the code.

2.1.2.4 extractProperties( ) Method

Example 2–5 shows a user-defined method named extractProperties( ), which sets the properties in the application object.

Example 2–5 extractProperties( ) Method (Audio)

```
public void extractProperties(OracleConnection con){
     byte[ ] ctx[ ] = new byte [4000][1];
     try {
          [1] Statement s = con.createStatement( );
          OracleResultSet rs = (OracleResultSet) s.executeQuery
               ("select * from TAUD where n = 1 for update");
          int index = 0;
          while(rs.next( )){
               index = rs.getInt(1);
               OrdAudio audObj = (OrdAudio) rs.getCustomDatum
                    (2, OrdAudio.getFactory( ));
               [2] audObj.setProperties(ctx);
               System.out.println("set Properties called");
               [3] if(audObj.checkProperties(ctx)){
                    System.out.println("checkProperties called");
                    System.out.println("setProperties successful");
                    System.out.println("checkProperties successful");
                    System.out.println("extraction  successful");
               }
               else{
                    System.out.println("checkProperties called");
                    System.out.println("extraction not successful");
                    System.out.println("checkProperties successful");
               }
               [4] OraclePreparedStatement stmt1 = (OraclePreparedStatement)
                    con.prepareCall("update taud set aud = ?
                    where n = " + index);
               stmt1.setCustomDatum(1,audObj);
               stmt1.execute( );
               stmt1.close( ) ;
               index++;
          }
          rs.close( );
          s.close( );
     }
     [5] catch(Exception e) {
          System.out.println("exception raised " + e);
          System.out.println("extract properties unsuccessful");
     }
}
```

The extractProperties( ) method performs the following operations:

1. Creates a statement, a local OracleResultSet, and a local ORDAudio object named audObj, and populates audObj with media data through the same process described in steps 1 through 5 of Example 2–4. In this method, you will be operating on the contents of the first row of the database table.
2. Calls setProperties( ) to extract properties values from the media data and set them in the application ORDAudio object. See "setProperties(byte[ ][ ])" in Chapter 3 for a list of the properties values extracted and set.
3. Calls checkProperties( ) to compare the properties values in the application object with the values in the media data. If all values are the same, checkProperties( ) returns TRUE and the appropriate messages are printed to the screen. If any values differ, checkProperties( ) returns FALSE and the appropriate messages are printed to the screen.
4. Creates and executes a SQL statement that will update the database ORDAudio object with the contents of audObj (including the properties extracted by setProperties( )).
5. Handles any errors or exceptions raised by the code.

2.1.2.5 printProperties( ) Method

Example 2–6 shows a user-defined method named printProperties( ), which prints the attributes of the application object to the screen.

Example 2–6 printProperties( ) Method (Audio)

```
public void printProperties(OracleConnection con){
     try {
         [1] Statement s = con.createStatement( );
          OracleResultSet rs = (OracleResultSet)
               s.executeQuery("select * from TAUD where n = 1 ");
          int index = 0;
          while(rs.next( )) {
               index = rs.getInt(1);
               OrdAudio audObj = (OrdAudio) rs.getCustomDatum
                    (2, OrdAudio.getFactory( ));
               [2] System.out.println("format: " + audObj.getFormat( ));
               System.out.println("mimeType: " + audObj.getMimeType( ));
               System.out.println("encoding: " + audObj.getEncoding( ));
               System.out.println("numberOfChannels: " +
                    audObj.getNumberOfChannels( ));
               System.out.println("samplingRate: " +
                    audObj.getSamplingRate( ));
               System.out.println("sampleSize: " + audObj.getSampleSize( ));
```

```
                System.out.println("compressionType : " +
                     audObj.getCompressionType( ));
                System.out.println("audioDuration: " +
                     audObj.getAudioDuration( ));
                System.out.println("description: " + audObj.getDescription( ));
           }
      }
      [3] catch(Exception e){
           System.out.println("exception raised " + e);
           System.out.println("print proerties unsuccessful");
      }
}
```

The printProperties( ) method performs the following operations:

1. Creates a statement, a local OracleResultSet, and a local ORDAudio object named audObj, and populates audObj with media data through the same process described in steps 1 through 5 of Example 2–4. In this method, you will be operating on the contents of the first row of the database table.
2. Gets the values of the properties in audObj and prints them to the screen.
3. Handles any errors or exceptions raised by the code.

2.1.2.6 otherMethods( ) Method

Example 2–7 shows a user-defined method named otherMethods( ), which attempts to use the processSourceCommand( ) method.

Example 2–7 otherMethods( ) Method (Audio)

```
public void otherMethods(OracleConnection con){
     byte[ ] ctx[ ] = {new byte[4000]};
     byte[ ] res[ ] = {new byte[20]};
     [1] int suc = 1;
     try {
          [2] Statement s1 = con.createStatement( );
          OracleResultSet rs1 = (OracleResultSet) s1.executeQuery
               ("select * from TAUD where n = 1 for update ");
          int index1 = 0;
          while(rs1.next( )) {
               index1 = rs1.getInt(1);
               OrdAudio audObj = (OrdAudio) rs1.getCustomDatum
                    (2, OrdAudio.getFactory( ));
               [3] try {
                    byte[ ] pSRes = audObj.processSourceCommand(ctx,
```

```
                    "", "", res);
                suc = 0;
            }
            [4] catch (Exception e) {
                System.out.println("Expected Exception raised in
                    processSourceCommand(...)" );
            }
            [5] OraclePreparedStatement stmt1 = (OraclePreparedStatement)
                con.prepareCall("update taud set aud = ? where
                n = " + index1);
            stmt1.setCustomDatum(1,audObj);
            stmt1.execute( );
            stmt1.close( ) ;
            index1++;
        }
        rs1.close( );
        s1.close( );
    }
    [6] catch(Exception e){
        System.out.println("Exception raised " );
    }
    [7] if(suc == 1)
        System.out.println("other methods successful");
    else
        System.out.println("other methods unsuccessful");
}
```

The otherMethods( ) method performs the following operations:

1. Creates an integer that will be used to indicate the success or failure of the method and sets it initially to 1 (for success).
2. Creates a statement, a local OracleResultSet, and a local ORDAudio object named audObj, and populates audObj with media data through the same process described in steps 1 through 5 of Example 2–4. In this method, you will be operating on the contents of the first row of the database table.
3. Tries to call processSourceCommand( ) with no value specified for the command to be called on the server side. This should raise an exception, which means the code following the processSourceCommand( ) call will not be run and the code in the catch loop will. If an exception is not raised, then the method has failed and the success indicator is set to 0 (for failure).
4. Prints the expected exception that was raised in step 3.

5. Creates and executes a SQL statement that will update the database ORDAudio object with the contents of audObj.
6. Handles any unexpected errors or exceptions raised by the code.
7. Prints the appropriate message to the screen based on the success or failure of the method.

2.1.2.7 loadDataFromStream( ) Method

Example 2–8 shows a user-defined method named loadDataFromStream( ), which uses the *inter*Media loadDataFromInputStream( ) method to load media data into the application object.

Example 2–8 loadDataFromStream( ) Method (Audio)

```
public void loadDataFromStream(OracleConnection con){
     try {
          [1] Statement s = con.createStatement( );
          OracleResultSet rs = (OracleResultSet) s.executeQuery
               ("select * from TAUD where n = 2 for update ");
          int index = 0;
          while(rs.next( )){
               index = rs.getInt(1);
               OrdAudio audObj = (OrdAudio) rs.getCustomDatum
                    (2, OrdAudio.getFactory( ));
               [2] FileInputStream fStream = new
                    FileInputStream("testaud.dat");
               [3] audObj.loadDataFromInputStream(fStream);
               [4] audObj.getDataInFile("output2.dat");
               [5] fStream.close( );
               System.out.println("************AFTER getDataInFile ");
               [6] System.out.println(" getContentLength output : " +
                    audObj.getContentLength( ));
               [7] OraclePreparedStatement stmt1 = (OraclePreparedStatement)
                    con.prepareCall("update taud set aud = ? where
                    n = " + index);
               stmt1.setCustomDatum(1,audObj);
               stmt1.execute( );
               stmt1.close( );
               index++;
          }
          System.out.println("load data from stream successful");
     }
     [8] catch(Exception e) {
```

```
            System.out.println("exception raised " + e);
            System.out.println("load data from stream unsuccessful");
        }
}
```

The loadDataFromStream( ) method performs the following operations:

1. Creates a statement, a local OracleResultSet, and a local ORDAudio object named audObj, and populates audObj with media data through the same process described in steps 1 through 5 of Example 2–4. In this method, you will be operating on the contents of the second row of the database table.
2. Creates a new FileInputStream object. This input stream contains the contents of the local file testaud.dat.
3. Uses the loadDataFromInputStream( ) method to load the media data in the input stream into the database ORDAudio object and into audObj. This also sets the local field on audObj, but not the database object.
4. Uses the getDataInFile( ) method to get the media data from the application ORDAudio object and load it into a file on the local system named output2.dat.
5. Closes the local input stream.
6. Gets the content length of audObj and prints it to the screen to verify the success of the loading.
7. Creates and executes a SQL statement that will update the database ORDAudio object with the contents of audObj. This update will set the attributes on the database object to match the application object.
8. Handles any errors or exceptions raised by the code.

2.1.2.8 loadDataFromByteArray( ) Method

Example 2–9 shows a user-defined method named loadDataFromByteArray( ), which uses the *inter*Media loadDataFromByteArray( ) method to load media data into the application object.

Example 2–9 loadDataFromByteArray( ) Method (Audio)

```
public void loadDataFromByteArray(OracleConnection con){
     try {
          [1] Statement s = con.createStatement( );
          OracleResultSet rs = (OracleResultSet) s.executeQuery
               ("select * from TAUD where n = 3 for update ");
          int index = 0;
```

```
while(rs.next( )) {
     index = rs.getInt(1);
     OrdAudio audObj = (OrdAudio) rs.getCustomDatum
          (2, OrdAudio.getFactory( ));
     [2] File ff = new File("testaud.dat");
     int fileLength = (int) ff.length( );
     byte[ ] data = new byte[fileLength];
     [3] FileInputStream fStream = new
          FileInputStream("testaud.dat");
     [4] fStream.read(data,0,fileLength);
     [5] audObj.loadDataFromByteArray(data);
     [6] fStream.close( );
     [7] audObj.getDataInFile("output3.dat");
     [8] byte[ ] resArr = audObj.getDataInByteArray( );
     [9] System.out.println("byte array length : " +
          resArr.length);
     [10] FileOutputStream outStream = new FileOutputStream
          ("output4.dat");
     [11] outStream.write(resArr);
     [12] outStream.close( );
     [13] InputStream inpStream = audObj.getDataInStream( );
     int length = 32300;
     byte[ ] tempBuffer = new byte[32300];
     [14] int numRead = inpStream.read(tempBuffer,0,length);
     try {
          [15] outStream = new FileOutputStream("output5.dat");
          [16] while (numRead != -1) {
               [17] if (numRead < 32300) {
                    length = numRead;
                    outStream.write(tempBuffer,0,length);
                    break;
               }
               [18] else
                    outStream.write(tempBuffer,0,length);
               [19] numRead = inpStream.read(tempBuffer,0,length);
          }
     }
     [20] finally {
          outStream.close( );
          inpStream.close( );
     }
     System.out.println("***********AFTER getDataInFile ");
     [21] System.out.println("getContentLength output : " +
          audObj.getContentLength( ));
     [22] OraclePreparedStatement stmt1 = (OraclePreparedStatement)
```

```
                    con.prepareCall("update taud set aud = ? where
                    n = " + index);
               stmt1.setCustomDatum(1,audObj);
               stmt1.execute( );
               stmt1.close( ) ;
               index++;
          }
     }
     [23] catch(Exception e) {
          System.out.println("exception raised " + e);
          System.out.println("load data from byte array unsuccessful");
    }
}
```

The loadDataFromByteArray( ) method performs the following operations:

1. Creates a statement, a local OracleResultSet, and a local ORDAudio object named audObj, and populates audObj with media data through the same process described in steps 1 through 5 of Example 2–4. In this method, you will be operating on the contents of the third row of the database table.
2. Determines the size (in bytes) of the local file testaud.dat and creates a byte array of the same size.
3. Creates a new FileInputStream object. This input stream contains the contents of testaud.dat.
4. Reads the contents of the input stream into the byte array.
5. Uses the loadDataFromByteArray( ) method to load the media data in the byte array into the database ORDAudio object and into audObj. This also sets the local field on audObj, but not the database object.
6. Closes the input stream.
7. Uses the getDataInFile( ) method to get the media data from the application ORDAudio object and load it into a file on the local system named output3.dat.
8. Uses the getDataInByteArray( ) method to get the media data from the application ORDAudio object and load it into a local byte array named resArr.
9. Gets the length of resArr and prints it to the screen to verify the success of the loading.
10. Creates a new FileOutputStream object named outStream. This output stream will write data to a local file named output4.dat.
11. Writes the contents of resArr to output4.dat.

12. Closes the output stream.
13. Creates a new input stream named inpStream. Uses the getDataInStream( ) method to get the media data from the application ORDAudio object and store it in inpStream.
14. Reads 32300 bytes from the beginning (that is, at an offset of 0) of inpStream into the byte array tempBuffer. The integer numRead will be set to the total number of bytes read, or -1 if the end of the input stream has been reached. In this case, if loading is successful, numRead should be equal to 32300.
15. Re-opens OutStream. In this case, it will write data to a local file named output5.dat.
16. Runs the operations in the while loop if numRead is not equal to -1. The program should enter this loop.
17. Enters the if loop if numRead is less than 32300 (that is, if not all the data was read). The if loop will write the number of bytes read into tempBuffer into outStream, and then break out of the loop.
18. Writes 32300 bytes into outStream if numRead is 32300.
19. Attempts to read more data from the input stream into the byte array. If 32300 bytes of data are read successfully, then numRead will be set to -1 and the program will exit the loop. If there is still unread data in the input stream, then it will be read into the byte array and steps 17 and 18 will be repeated.
20. Closes both the input stream and the output stream after exiting the while loop.
21. Gets the content length of audObj and prints it to the screen to verify the success of the loading.
22. Creates and executes a SQL statement that will update the database ORDAudio object with the contents of audObj. This update will set the attributes on the database object to match the application object.
23. Handles any errors or exceptions raised by the code.

2.2 Image Example

The image example (including ImageExample.sql and ImageExample.java) contains user-defined methods that use SQL, JDBC, and *inter*Media Java Classes APIs to perform the following operations:

- Create a database server table that contains test content

- Load data into both application and database ORDImage objects from a local file and set the local field on both the application and database objects
- Load data into both application and database ORDImage objects from a local stream and set the local field on both the application and database objects
- Load data into both application and database ORDImage objects from a local byte array and set the local field on both the application and database objects
- Extract and print properties from the application ORDImage object
- Show an example of the process( ) and processCopy( ) methods

2.2.1 ImageExample.sql

Example 2–10 shows the contents of ImageExample.sql.

Example 2–10 Contents of ImageExample.sql

```
set echo on

-- Please Change system password.
connect system/manager
drop user IMAGEUSER cascade;

[1] grant connect,resource to IMAGEUSER identified by IMAGEUSER;

-- Replace C:\Oracle\Ora81' with your ORACLE HOME
[2] create or replace directory ORDIMAGEDIR as 'C:\Oracle\Ora81\ord\img\demo';
grant read on directory ORDIMAGEDIR to public with grant option;

[3] connect IMAGEUSER/IMAGEUSER;

[4] create table ordimagetab(id number, image ORDSYS.ORDImage, image2
ORDSYS.ORDImage);

-- Note - the ORDImage.init method was added in interMedia 8.1.7.
-- If you are running against an older releasen of interMedia and Oracle,
-- you will have to modify the following INSERT statements to use the
-- ORDImage default constructor.
--
[5] insert into ordimagetab values
(1, ORDSYS.ORDImage.init( ),
    ORDSYS.ORDImage.init( ));

insert into ordimagetab values
```

```
(2, ORDSYS.ORDImage.init( ),
    ORDSYS.ORDImage.init( ));

insert into ordimagetab values
(3, ORDSYS.ORDImage.init( ),
    ORDSYS.ORDImage.init( ));

insert into ordimagetab values
(4, ORDSYS.ORDImage.init( ),
    ORDSYS.ORDImage.init( ));

[6] insert into ordimagetab values
(5, ORDSYS.ORDImage.init('file','ORDIMAGEDIR','imgdemo.dat'),
    ORDSYS.ORDImage.init( ));

insert into ordimagetab values
(6, ORDSYS.ORDImage.init('file','ORDIMAGEDIR','imgdemo.dat'),
    ORDSYS.ORDImage.init( ));

commit;
set echo off
exit;
```

The SQL statements in ImageExample.sql perform the following operations:

1. Create a user named IMAGEUSER and grant the appropriate permissions to the user.
2. Create a directory named ORDIMAGEDIR and set the appropriate permissions. You will need to change the directory to match your schema.
3. Connect to the database server as IMAGEUSER.
4. Create a table named ordimagetab, which contains one column of numbers and two columns of ORDImage objects.
5. Using the init method, add four rows with two empty objects each.
6. Using the init method, add two rows with one object with values and one empty object.

The ORDImage.init method was added in release 8.1.7. If you are running against a previous release of *inter*Media and Oracle8*i*, you will have to modify the INSERT statements in steps 5 and 6 to use the ORDImage default constructor.

See *Oracle8i interMedia Audio, Image, and Video User's Guide and Reference* for more information on the init method.

2.2.2 ImageExample.java

Section 2.2.2.1 through Section 2.2.2.9 show the methods contained in the ImageExample.java sample file.

2.2.2.1 main( ) Method

Example 2–11 shows the main( ) method.

Example 2–11 main( ) Method (Image)

```
public static void main (String args[ ]){
     byte[ ] ctx = new byte[4000];
     OracleConnection con = null;
     try{
          ImageExample ie = new ImageExample( );
          [1] con = ie.connect( );
          //Include the following line only if you are running
          //an Oracle 8.1.7 database or later.
          //If you are running a database server prior to 8.1.7,
          //the call will fail.
          [2] OrdMediaUtil.imCompatibilityInit(con);
          [3] ie.setPropertiesExample(con);
          ie.displayPropertiesExample(con);
          ie.fileBasedExample(con);
          ie.streamBasedExample(con);
          ie.byteArrayBasedExample(con);
          ie.processExample(con);
          [4] con.commit( );
          [5] con.close( );
          System.out.println("Done.");
     }
     [6] catch (Exception e){
          try{
               System.out.println("Exception : " + e);
               con.close( );
          }
          catch(Exception ex){
               System.out.println("Close Connection Exception : " + ex);
          }
     }
}
```

The code in the main( ) method performs the following operations:

1. Uses the connect( ) method to make a connection to a database table.

2. Ensures the compatibility of your 8.1.7 application; this will only work if your database server is at least release 8.1.7. See Section 1.7 for more information.
3. Calls several methods (also defined in ImageExample.java) that manipulate objects on the database server and the local machine.
4. Commits any changes made to the database table.
5. Closes the connection to the database.
6. Handles any errors or exceptions raised by the code.

Section 2.2.2.2 through Section 2.2.2.9 will provide information on the methods called from the main( ) method.

2.2.2.2 connect( )Method

Example 2–12 shows a user-defined method named connect( ), which makes a connection from the application to the database.

Example 2–12 connect( ) Method (Image)

```
public OracleConnection connect( ) throws Exception{
     String connectString;
     [1] Class.forName ("oracle.jdbc.driver.OracleDriver");
     [2] connectString = "jdbc:oracle:oci8:@";
     [3] OracleConnection con = (OracleConnection)DriverManager.getConnection
          (connectString,"IMAGEUSER","IMAGEUSER");
     [4] con.setAutoCommit(false);
     return con;
}
```

The connect( ) method performs the following operations:

1. Loads the JDBC drivers directly, because Oracle uses a JDK-compliant Java virtual machine.
2. Defines a string that contains the URL of the database to which you will connect. You may need to change this string to match your database.
3. Sets the connection to the database, using the URL contained in connectString, the user name IMAGEUSER, and the password IMAGEUSER. The user name and password were created by ImageExample.sql.
4. Disables the auto-commit mode. This means that you must commit or roll back manually with the commit( ) or rollback( ) methods, respectively.

2.2.2.3 setPropertiesExample( ) Method

Example 2–13 shows a user-defined method named setPropertiesExample( ), which sets the properties in the application object.

Example 2–13 setPropertiesExample( ) Method (Image)

```
public void setPropertiesExample(OracleConnection con){
     try{
          int index = 0;
          [1] Statement s = con.createStatement( );
          [2] OracleResultSet rs = (OracleResultSet)s.executeQuery
               ("select * from ordimagetab where id = 5 for update");
          [3] while(rs.next( )){
               [4] index = rs.getInt(1);
               [5] OrdImage imgObj = (OrdImage)rs.getCustomDatum
                    (2, OrdImage.getFactory( ));
               [6] imgObj.setProperties( );
               System.out.println("set Properties called");
               [7] if(imgObj.checkProperties( )){
                    System.out.println("checkProperties called");
                    System.out.println("setProperties successful");
                    System.out.println("checkProperties successful");
                    System.out.println("successful");
               }
               else{
                    System.out.println("checkProperties called");
                    System.out.println("setProperties not successful");
                    System.out.println("checkProperties successful");
               }
               [8] OraclePreparedStatement stmt1 = (OraclePreparedStatement)
                    con.prepareCall("update ordimagetab set
                    image = ? where id = " + index);
               stmt1.setCustomDatum(1,imgObj);
               stmt1.execute( );
               stmt1.close( ) ;
          }
          rs.close( );
          s.close( );
     }
     [9] catch(Exception e){
          System.out.println("exception raised " + e);
     }
}
```

The setPropertiesExample( ) method performs the following operations:

1. Creates an OracleStatement object.
2. Executes the given SQL query and puts the results into a local OracleResultSet object.
3. Performs the operations in the loop while there are results in the OracleResultSet that have not been processed. However, in this case, there is only one row included in the OracleResultSet, so the operations in the loop will run once.
4. Sets an index variable to the value of the integer in the first column of the first row in the OracleResultSet (in this case, the value is 5).
5. Creates a local ORDImage object named imgObj. Populates imgObj with the contents of the ORDImage object in the second column of the current row in the OracleResultSet.
6. Calls setProperties( ) to extract properties values from the media data and set them in the application ORDImage object. See "setProperties( )" in Chapter 4 for a list of the properties values extracted and set.
7. Calls checkProperties( ) to compare the properties values in the application object with the values in the media data. If all values are the same, checkProperties( ) returns TRUE and the appropriate messages are printed to the screen. If any values differ, checkProperties( ) returns FALSE and the appropriate messages are printed to the screen.
8. Creates and executes a SQL statement that will update the database ORDImage object with the contents of imgObj.
9. Handles any errors or exceptions raised by the code.

2.2.2.4 displayPropertiesExample( ) Method

Example 2–14 shows a user-defined method named displayPropertiesExample( ), which prints the attributes of the application object to the screen.

Example 2–14 displayPropertiesExample( ) Method (Image)

```
public void displayPropertiesExample(OracleConnection con){
     try{
          int index = 0;
          [1] Statement s = con.createStatement( );
          OracleResultSet rs = (OracleResultSet)s.executeQuery(
               "select * from ordimagetab where id = 5 ");
```

```
            while(rs.next( )){
                 index = rs.getInt(1);
                 OrdImage imgObj = (OrdImage) rs.getCustomDatum(2,
                      OrdImage.getFactory( ));
                 [2] System.out.println("format : " + imgObj.getFormat( ));
                 System.out.println("mimeType: " + imgObj.getMimeType( ));
                 System.out.println("height: " + imgObj.getHeight( ));
                 System.out.println("width: " + imgObj.getWidth( ));
                 System.out.println("contentLength: " +
                      imgObj.getContentLength( ));
                 System.out.println("contentFormat: " +
                      imgObj.getContentFormat( ));
                 System.out.println("compressionFormat: " +
                      imgObj.getCompressionFormat( ));
                 System.out.println("source type: " +
                      imgObj.getSourceType( ));
                 System.out.println("source loc: " +
                      imgObj.getSourceLocation( ));
                 System.out.println("source name: " + imgObj.getSourceName( ));
                 System.out.println("source : " + imgObj.getSource( ));
                 [3] try{
                      String attrString = getAllAttributesAsString(imgObj);
                      System.out.println(attrString);
                 }
                 [4] catch (Exception e){
                      System.out.println("Exception raised in
                           getAllAttributesAsString:");
                 }
                 System.out.println("successful");
            }
       }
       [5] catch(Exception e) {
            System.out.println("exception raised " + e);
       }
}
```

The displayPropertiesExample( ) method performs the following operations:

1. Creates a statement, a local OracleResultSet, and a local ORDImage object named imgObj, and populates imgObj with media data through the same process described in steps 1 through 5 of Example 2–13. In this method, you will be operating on the contents of the fifth row of the database table. This is the same row you operated on in Example 2–13.
2. Gets the values of the properties in imgObj and prints them to the screen.

3. Gets the attributes of imgObj and stores them in a string by using the getAllAttributesAsString( ) method, and prints the contents of the string to the screen. See Section 2.2.2.5 for more information on getAllAttributesAsString( ).
4. Handles any errors or exceptions raised by the call to getAllAttributesAsString( ).
5. Handles any errors or exceptions raised by the code in general.

2.2.2.5 getAllAttributesAsString( ) Method

Example 2–15 shows a user-defined method named getAllAttributesAsString( ), which creates a String object that contains the values of the application object attributes.

Example 2–15 getAllAttributesAsString( ) Method (Image)

```
public String getAllAttributesAsString (OrdImage imgObj) throws Exception{
     [1] String attStr = imgObj.getSource( ) + " mimeType = " +
          imgObj.getMimeType( ) + ", fileFormat = " +
          imgObj.getFormat( ) + ", height = " + imgObj.getHeight( )
          + ", width = " + imgObj.getWidth( ) + ", contentLength = "
          + imgObj.getContentLength( ) + ", contentFormat = " +
          imgObj.getContentFormat( ) + ", compressionFormat = " +
          imgObj.getCompressionFormat( );
     [2] return attStr;
}
```

The getAllAttributesAsString( ) method performs the following operations:

1. Creates a String object named attStr. Gets the values of several attributes from the application image object and stores their values in attStr.
2. Returns attStr to the method that called this method.

2.2.2.6 fileBasedExample( ) Method

Example 2–16 shows a user-defined method named fileBasedExample( ), which uses the loadDataFromFile( ) method to load media data into the application object.

Example 2–16 fileBasedExample( ) Method (Image)

```
public void fileBasedExample(OracleConnection con){
     try{
          int index = 0;
          [1] Statement s = con.createStatement( );
```

```
            OracleResultSet rs = (OracleResultSet)s.executeQuery(
                 "select * from ORDIMAGETAB where id = 2 for update ");
            while(rs.next( )){
                 index = rs.getInt(1);
                 OrdImage imgObj = (OrdImage) rs.getCustomDatum(2,
                      OrdImage.getFactory( ));
                 [2] imgObj.loadDataFromFile("imgdemo.dat");
                 [3] imgObj.setProperties( );
                 [4] imgObj.getDataInFile("fileexample.dat");
                 [5] OraclePreparedStatement stmt1 = (OraclePreparedStatement)
                      con.prepareCall("update ordimagetab set image =
                      ? where id = " + index);
                 stmt1.setCustomDatum(1,imgObj);
                 stmt1.execute( );
                 stmt1.close( ) ;
            }
            System.out.println("successful");
       }
       [6] catch(Exception e){
            System.out.println("exception raised " + e);
       }
}
```

The fileBasedExample( ) method performs the following operations:

1. Creates a statement, a local OracleResultSet, and a local ORDImage object named imgObj, and populates imgObj with media data through the same process described in steps 1 through 5 of Example 2–13. In this method, you will be operating on the contents of the second row of the database table.
2. Uses the loadDataFromFile( ) method to load the media data from the local file imgdemo.dat into the database ORDImage object and into imgObj. This also sets the local field on imgObj, but not the database object.
3. Calls setProperties( ) to extract properties values from the media data and set them in the application ORDImage object. See "setProperties( )" in Chapter 4 for a list of the properties values extracted and set.
4. Uses the getDataInFile( ) method to get the media data from the application ORDImage object and loads it into a file on the local system named fileexample.dat.
5. Creates and executes a SQL statement that will update the database ORDImage object with the contents of imgObj. This update will set the attributes on the database object, to match the application object.

6. Handles any errors or exceptions raised by the code.

2.2.2.7 streamBasedExample( ) Method

Example 2–17 shows a user-defined method named streamBasedExample( ), which uses the loadDataFromInputStream( ) method to load media data into the application object.

Example 2–17 streamBasedExample( ) Method (Image)

```
public void streamBasedExample(OracleConnection con){
     try{
          int index = 0;
          [1] Statement s = con.createStatement( );
          OracleResultSet rs = (OracleResultSet)s.executeQuery(
               "select * from ORDIMAGETAB where id = 3 for update ");
          while(rs.next( )){
               index = rs.getInt(1);
               OrdImage imgObj = (OrdImage) rs.getCustomDatum(2,
                    OrdImage.getFactory( ));
               [2] FileInputStream fStream = new FileInputStream
                    ("imgdemo.dat");
               [3] imgObj.loadDataFromInputStream(fStream);
               [4] fStream.close( );
               [5] imgObj.setProperties( );
               [6] InputStream inpStream = imgObj.getDataInStream( );
               int length = 32300;
               byte[ ] tempBuffer = new byte[length];
               [7] int numRead = inpStream.read(tempBuffer,0,length);
               FileOutputStream outStream=null;
               try{
                    [8] outStream = new FileOutputStream
                         ("streamexample.dat");
                    [9] while(numRead != -1){
                         [10] if (numRead < length){
                              length = numRead;
                              outStream.write(tempBuffer,0,length);
                              break;
                         }
                         [11] else
                              outStream.write(tempBuffer,0,length);
                         [12] numRead = inpStream.read(tempBuffer,0,
                              length);
                    }
               }
```

```
                [13] finally{
                     if (outStream != null)
                          outStream.close( );
                     inpStream.close( );
                }
                [14] OraclePreparedStatement stmt1 = (OraclePreparedStatement)
                     con.prepareCall("update ordimagetab set
                     image = ? where id = " + index);
                stmt1.setCustomDatum(1,imgObj);
                stmt1.execute( );
                stmt1.close( ) ;
           }
           System.out.println("successful");
      }
      [15] catch(Exception e){
           System.out.println("exception raised " + e);
      }
}
```

The streamBasedExample( ) method performs the following operations:

1. Creates a statement, a local OracleResultSet, and a local ORDImage object named imgObj, and populates imgObj with media data through the same process described in steps 1 through 5 of Example 2–13. In this method, you will be operating on the contents of the third row of the database table.
2. Creates a new FileInputStream object. This input stream contains the contents of the local file imgdemo.dat.
3. Uses the loadDataFromInputStream( ) method to load the media data in the input stream into the database ORDImage object and into imgObj. This also sets the local field on imgObj, but not the database object.
4. Closes the input stream.
5. Calls setProperties( ) to extract properties values from the media data and set them in the application ORDImage object. See "setProperties( )" in Chapter 4 for a list of the properties values extracted and set.
6. Creates a new InputStream named inpStream. Calls getDataInStream( ) to get the media data from the application ORDImage object and stores it in inpStream.
7. Reads 32300 bytes from the beginning (that is, at an offset of 0) of inpStream into the byte array tempBuffer. The integer numRead will be set to the total

number of bytes read, or -1 if the end of the input stream has been reached. In this case, if loading is successful, numRead should be equal to 32300.

8. Creates a new FileOutputStream object named outStream. This output stream will write data to a local file named streamexample.dat.
9. Runs the operations in the while loop if numRead is not equal to -1. The program should enter this loop.
10. Writes the number of bytes read into tempBuffer into outStream if numRead is less than 32300 (that is, if not all the data was read).
11. Writes 32300 bytes into outStream if numRead is 32300.
12. Attempts to read more data from the input stream into the byte array. If 32300 bytes of data are read successfully, then numRead will be set to -1 and the program will exit the loop. If there is still unread data in the input stream, then it will be read into the byte array and steps 10 and 11 will be repeated.
13. Closes both the input stream and the output stream after exiting the while loop.
14. Creates and executes a SQL statement that will update the database ORDImage object with the contents of imgObj. This update will set the attributes on the database object to match the application object.
15. Handles any errors or exceptions raised by the code.

2.2.2.8 byteArrayBasedExample( ) Method

Example 2–18 shows a user-defined method named byteArrayBasedExample( ), which uses the loadDataFromByteArray( ) method to load media data into the application object.

Example 2–18 byteArrayBasedExample( ) Method (Image)

```
public void byteArrayBasedExample(OracleConnection con){
     try{
          int index = 0;
          [1] Statement s = con.createStatement( );
          OracleResultSet rs = (OracleResultSet)s.executeQuery
               ("select * from ORDIMAGETAB where id = 4 for update ");
          while(rs.next( )){
               index = rs.getInt(1);
               OrdImage imgObj = (OrdImage) rs.getCustomDatum(2,
                    OrdImage.getFactory( ));
               [2] File ff = new File("imgdemo.dat");
               int fileLength = (int) ff.length( );
```

```
                byte[ ] data = new byte[fileLength];
                [3] FileInputStream fStream = new
                     FileInputStream("imgdemo.dat");
                [4] fStream.read(data,0,fileLength);
                [5] imgObj.loadDataFromByteArray(data);
                [6] fStream.close( );
                [7] imgObj.setProperties( );
                [8] byte[ ] resArr = imgObj.getDataInByteArray( );
                [9] System.out.println("byte array length : " +
                     resArr.length);
                [10] OraclePreparedStatement stmt1 = (OraclePreparedStatement)
                     con.prepareCall("update ordimagetab set image =
                     ? where id = " + index);
                stmt1.setCustomDatum(1,imgObj);
                stmt1.execute( );
                stmt1.close( ) ;
           }
           System.out.println("successful");
      }
      [11] catch(Exception e){
           System.out.println("exception raised " + e);
      }
}
```

The byteArrayBasedExample( ) method performs the following operations:

1. Creates a statement, a local OracleResultSet, and a local ORDImage object named imgObj, and populates imgObj with media data through the same process described in steps 1 through 5 of Example 2–13. In this method, you will be operating on the contents of the fourth row of the database table.
2. Determines the size (in bytes) of the local file imgdemo.dat and creates a byte array of the same size.
3. Creates a new FileInputStream object. This input stream contains the contents of imgdemo.dat.
4. Reads the contents of the input stream into the byte array.
5. Uses the loadDataFromByteArray( ) method to load the media data in the byte array into the database ORDImage object and into imgObj. This also sets the local field on imgObj, but not the database object.
6. Closes the input stream.

7. Calls setProperties( ) to extract properties values from the media data and set them in the application ORDImage object. See "setProperties( )" in Chapter 4 for a list of the properties values extracted and set.
8. Uses the getDataInByteArray( ) method to get the media data from the application ORDImage object and load it into a local byte array named resArr.
9. Gets the length of resArr and prints it to the screen to verify the success of the loading.
10. Creates and executes a SQL statement that will update the database ORDImage object with the contents of imgObj. This update will set the attributes on the database object to match the application object.
11. Handles any errors or exceptions raised by the code.

2.2.2.9 processExample( ) Method

Example 2–19 shows a user-defined method named processExample( ), which uses the process( ) and processCopy( ) methods to manipulate the media data in the application object.

Example 2–19 processExample( ) Method (Image)

```
public void processExample(OracleConnection con){
     try{
          int index1 = 0;
          [1] Statement s1 = con.createStatement( );
          OracleResultSet rs1 = (OracleResultSet)s1.executeQuery
               ("select * from ORDIMAGETAB where id = 2 for update ");
          while(rs1.next( )){
               index1 = rs1.getInt(1);
               OrdImage imgObj = (OrdImage) rs1.getCustomDatum(2,
                    OrdImage.getFactory( ));
               [2] OrdImage imgObj2 = (OrdImage) rs1.getCustomDatum(3,
                    OrdImage.getFactory( ));
               try{
                    [3] imgObj.processCopy("maxScale=32 32, fileFormat=
                         GIFF", imgObj2);
                    [4] imgObj.process("fileFormat=JFIF");
                    [5] System.out.println(getAllAttributesAsString
                         (imgObj));
                    [6] System.out.println(getAllAttributesAsString(imgObj2));
               }
               [7] catch (Exception e){
                    System.out.println("Exception raised in process"
```

```
                    + e );
               }
               [8] OraclePreparedStatement stmt1 = (OraclePreparedStatement)
                    con.prepareCall("update ordimagetab set image =
                    ?, image2 = ? where id = " + index1);
               stmt1.setCustomDatum(1,imgObj);
               stmt1.setCustomDatum(2,imgObj2);
               stmt1.execute( );
               stmt1.close( ) ;
          }
          rs1.close( );
          s1.close( );
     }
     [9] catch(Exception e){
          System.out.println("Exception raised: " + e);
     }
     System.out.println("successful");
}
```

The processExample( ) method performs the following operations:

1. Creates a statement, a local OracleResultSet, and a local ORDImage object named imgObj, and populates imgObj with media data through the same process described in steps 1 through 5 of Example 2–13. In this method, you will be operating on the contents of the second row of the database table. The database ORDImage object is named image.
2. Creates a local ORDImage object named imgObj2. Populates imgObj2 with the contents of the ORDImage object in the third column of the current row in the OracleResultSet. This database ORDImage column is named image2.
3. Populates the image data in imgObj2 with a 32 x 32 GIF thumbnail image generated from the image data in imgObj. imgObj is unchanged by this operation.
4. Uses the process( ) method to convert the image in imgObj to a JPEG (JFIF) image.
5. Gets the attributes of imgObj by using the getAllAttributesAsString( ) method, and prints the attributes to the screen. See Section 2.2.2.5 for more information on getAllAttributesAsString( ).
6. Gets the attributes of imgObj2 by using the getAllAttributesAsString( ) method, and prints the attributes to the screen. See Section 2.2.2.5 for more information on getAllAttributesAsString( ).

7. Handles any errors or exceptions raised by the code in steps 3 through 6.
8. Creates and executes a SQL statement that will update the appropriate database ORDImage objects with the contents of imgObj and imgObj2.
9. Handles any errors or exceptions raised by the code.

2.3 Video Example

The video example (including VideoExample.sql and VideoExample.java) contains user-defined methods that use SQL, JDBC, and *inter*Media Java Classes APIs to perform the following operations:

- Create a database server table that contains test content
- Load data into both application and database ORDVideo objects from a local file and set the local field on both the application and database objects
- Load data into both application and database ORDVideo objects from a local stream and set the local field on both the application and database objects
- Load data into both application and database ORDVideo objects from a local byte array and set the local field on both the application and database objects
- Extract and print properties from the application ORDVideo object
- Demonstrate error handling through a failed call to a database method

2.3.1 VideoExample.sql

Example 2–20 shows the contents of VideoExample.sql.

Example 2–20 Contents of VideoExample.sql

```
set echo on

--PLEASE change system password
connect system/manager
drop user VIDEOUSER cascade;
[1] create user VIDEOUSER identified by VIDEOUSER ;
grant connect,resource to VIDEOUSER identified by VIDEOUSER;

[2] connect VIDEOUSER/VIDEOUSER

[3] CREATE TABLE TVID(n NUMBER, vid ORDSYS.ORDVIDEO);
```

```
-- Note - the ORDVideo.init method was added in interMedia 8.1.7.
-- If you are running against an older release of interMedia and Oracle,
-- you will have to modify the following INSERT statements to use the
-- ORDVideo default constructor.

[4] INSERT INTO TVID VALUES(1, ORDSYS.ORDVideo.init( ));
INSERT INTO TVID VALUES(2, ORDSYS.ORDVideo.init( ));
INSERT INTO TVID VALUES(3, ORDSYS.ORDVideo.init( ));
commit;
/
```

The SQL statements in VideoExample.sql perform the following operations:

1. Create a user named VIDEOUSER and connect as VIDEOUSER.
2. Connect to the database server as VIDEOUSER.
3. Create a table named TVID with two columns: a column of numbers and a column of ORDVideo objects.
4. Add three rows to the table, each containing an empty ORDVideo object.

The ORDVideo.init method was added in release 8.1.7. If you are running against a previous release of *inter*Media and Oracle, you will have to modify the INSERT statements in step 4 to use the ORDVideo default constructor.

See *Oracle8i interMedia Audio, Image, and Video User's Guide and Reference* for more information on the init method.

2.3.2 VideoExample.java

Section 2.3.2.1 through Section 2.3.2.8 show the methods contained in the VideoExample.java sample file.

2.3.2.1 main( ) Method

Example 2–21 shows the main( ) method.

Example 2–21 main( ) Method (Video)

```
public static void main (String args[ ]){
     byte[ ] ctx = new byte[4000];
     OracleConnection con = null;
     try {
          VideoExample tk = new VideoExample( );
          [1] con = tk.connect( );
          //Include the following line only if you are running
```

```
            //an Oracle 8.1.7 database or later.
            //If you are running a database server prior to 8.1.7,
            //the call will fail.
            [2] OrdMediaUtil.imCompatibilityInit(con);
            [3] tk.loadDataFromFile(con);
            tk.extractProperties(con);
            tk.printProperties(con);
            tk.loadDataFromStream(con);
            tk.otherMethods(con);
            tk.loadDataFromByteArray(con);
            [4] con.commit( );
            [5] con.close( );
            System.out.println("Done.");
        }
        [6] catch (Exception e) {
            try {
                System.out.println("Exception : " + e);
                con.close( );
            }
            catch(Exception ex) {
                System.out.println("Close Connection Exception : " + ex);
            }
        }
}
```

The code in the main( ) method performs the following operations:

1. Uses the connect( ) method to make a connection to a database table.
2. Ensures the compatibility of your 8.1.7 application; this will only work if your database server is at least release 8.1.7. See Section 1.7 for more information.
3. Calls several methods (also defined in VideoExample.java) that manipulate objects on the database server and the local machine.
4. Commits any changes made to the database table.
5. Closes the connection to the database.
6. Handles any errors or exceptions raised by the code.

Section 2.3.2.2 through Section 2.3.2.8 will provide information on the methods called from the main( ) method in the order in which they are called, not in the order they appear in VideoExample.java.

2.3.2.2 connect( ) Method

Example 2–22 shows a user-defined method named connect( ), which makes a connection from the application to the database.

Example 2–22 connect( ) Method (Video)

```
public OracleConnection connect( ) throws Exception{
     String connectString;
     [1] Class.forName ("oracle.jdbc.driver.OracleDriver");
     [2] connectString = "jdbc:oracle:oci8:@";
     [3] OracleConnection con = (OracleConnection)
          DriverManager.getConnection(connectString,"VIDEOUSER","VIDEOUSER");
     [4] con.setAutoCommit(false);
     return con;
}
```

The connect( ) method performs the following operations:

1. Loads the JDBC drivers directly, because Oracle uses a JDK-compliant Java virtual machine.
2. Defines a string that contains the URL of the database to which you will connect. You may need to change this string to match your database.
3. Sets the connection to the database, using the URL contained in connectString, the user name VIDEOUSER, and the password VIDEOUSER. The user name and password were created by VideoExample.sql.
4. Disables the auto-commit mode. This means that you must commit or roll back manually with the commit( ) or rollback( ) methods, respectively.

2.3.2.3 loadDataFromFile( ) Method

Example 2–23 shows a user-defined method named loadDataFromFile( ), which uses the *inter*Media loadDataFromFile( ) method to load media data into the application object.

Example 2–23 loadDataFromFile( ) Method (Video)

```
public void loadDataFromFile(OracleConnection con){
     try {
          [1] Statement s = con.createStatement( );
          [2] OracleResultSet rs = (OracleResultSet)s.executeQuery
               ("select * from TVID where n = 1 for update ");
          int index = 0;
          [3] while(rs.next( )){
```

```
                [4] index = rs.getInt(1);
                [5] OrdVideo vidObj = (OrdVideo) rs.getCustomDatum(2,
                     OrdVideo.getFactory( ));
                [6] vidObj.loadDataFromFile("testvid.dat");
                [7] vidObj.getDataInFile("output1.dat");
                System.out.println("************AFTER getDataInFile ");
                [8] System.out.println("getContentLength output : " +
                     vidObj.getContentLength( ));
                [9] OraclePreparedStatement stmt1 = (OraclePreparedStatement)
                     con.prepareCall("update tvid set vid = ?
                     where n = " + index);
                stmt1.setCustomDatum(1,vidObj);
                stmt1.execute( );
                stmt1.close( ) ;
                index++;
           }
           System.out.println("loading successful");
      }
      [10] catch(Exception e) {
           System.out.println("exception raised " + e);
           System.out.println("loading unsuccessful");
      }
}
```

The loadDataFromFile( ) method performs the following operations:

1. Creates an OracleStatement object.
2. Executes the given SQL query and puts the results into a local OracleResultSet object.
3. Performs the operations in the loop while there are results in the OracleResultSet that have not been processed. However, in this case, there is only one row included in the OracleResultSet, so the operations in the loop will run once.
4. Sets an index variable to the value of the integer in the first column of the first row in the OracleResultSet (in this case, the value is 1).
5. Creates a local ORDVideo object named vidObj. Populates vidObj with the contents of the ORDVideo object in the second column of the current row in the OracleResultSet.
6. Uses the loadDataFromFile( ) method to load the media data in testvid.dat into the database ORDVideo object and into vidObj. This also sets the local field on vidObj, but not the database object.

7. Uses the getDataInFile( ) method to get the media data from the application ORDVideo object and loads it into a file on the local system named output1.dat.
8. Gets the content length of vidObj and prints it to the screen to verify the success of the loading.
9. Creates and executes a SQL statement that will update the database ORDVideo object with the contents of vidObj. This update will set the local attribute on the database object to match the application object.
10. Handles any errors or exceptions raised by the code.

2.3.2.4 extractProperties( ) Method

Example 2–24 shows a user-defined method named extractProperties( ), which sets the properties in the application object.

Example 2–24 extractProperties( ) Method (Video)

```
public void extractProperties(OracleConnection con){
     byte[ ] ctx[ ] = new byte [4000][1];
     try {
          [1] Statement s = con.createStatement( );
          OracleResultSet rs = (OracleResultSet)s.executeQuery
               ("select * from TVID where n = 1 for update");
          int index = 0;
          while(rs.next( )){
               index = rs.getInt(1);
               OrdVideo vidObj = (OrdVideo) rs.getCustomDatum(2,
                    OrdVideo.getFactory( ));
               [2] vidObj.setProperties(ctx);
               System.out.println("set Properties called");
               [3] if(vidObj.checkProperties(ctx)) {
                    System.out.println("checkProperties called");
                    System.out.println("setBindParams successful");
                    System.out.println("setProperties successful");
                    System.out.println("checkProperties successful");
                    System.out.println("extraction  successful");
               }
               else {
                    System.out.println("checkProperties called");
                    System.out.println("extraction not successful");
                    System.out.println("checkProperties successful");
               }
               [4] OraclePreparedStatement stmt1 = (OraclePreparedStatement)
                    con.prepareCall("update tvid set vid = ? where
```

```
                    n = " + index);
               stmt1.setCustomDatum(1,vidObj);
               stmt1.execute( );
               stmt1.close( ) ;
               index++;
          }
          rs.close( );
          s.close( );
     }
     [5] catch(Exception e) {
          System.out.println("exception raised " + e);
          System.out.println("extract prop unsuccessful");
     }
}
```

The extractProperties( ) method performs the following operations:

1. Creates a statement, a local OracleResultSet, and a local ORDVideo object named vidObj, and populates vidObj with media data through the same process described in steps 1 through 5 of Example 2–23. In this method, you will be operating on the contents of the first row of the database table.
2. Calls setProperties( ) to extract properties values from the media data and set them in the application ORDVideo object. See "setProperties(byte[ ][ ])" in Chapter 5 for a list of the properties values extracted and set.
3. Calls checkProperties( ) to compare the properties values in the application object with the values in the media data. If all values are the same, checkProperties( ) returns TRUE and the appropriate messages are printed to the screen. If any values differ, checkProperties( ) returns FALSE and the appropriate messages are printed to the screen.
4. Creates and executes a SQL statement that will update the database ORDVideo object with the contents of vidObj.
5. Handles any errors or exceptions raised by the code.

2.3.2.5 printProperties( ) Method

Example 2–25 shows a user-defined method named printProperties( ), which prints the attributes of the application object to the screen.

Example 2–25 printProperties( ) Method (Video)

```
public void printProperties(OracleConnection con){
     try {
```

```
            [1] Statement s = con.createStatement( );
            OracleResultSet rs = (OracleResultSet)s.executeQuery
                 ("select * from TVID where n = 1 ");
            int index = 0;
            while(rs.next( )){
                 index = rs.getInt(1);
                 OrdVideo vidObj = (OrdVideo) rs.getCustomDatum(2,
                      OrdVideo.getFactory( ));
                 [2] System.out.println("format: " + vidObj.getFormat( ));
                 System.out.println("mimetype: " + vidObj.getMimeType( ));
                 System.out.println("width: " + vidObj.getWidth( ));
                 System.out.println("height: " + vidObj.getHeight( ));
                 System.out.println("frame resolution: " +
                      vidObj.getFrameResolution( ));
                 System.out.println("frame rate: " + vidObj.getFrameRate( ));
                 System.out.println("video duration: " +
                      vidObj.getVideoDuration( ));
                 System.out.println("number of frames: " +
                      vidObj.getNumberOfFrames( ));
                 System.out.println("description : " +
                      vidObj.getDescription( ));
                 System.out.println("compression type: " +
                      vidObj.getCompressionType( ));
                 System.out.println("bit rate: " + vidObj.getBitRate( ));
                 System.out.println("num of colors: " +
                      vidObj.getNumberOfColors( ));
            }
       }
       [3] catch(Exception e) {
            System.out.println("exception raised " + e);
            System.out.println("print proerties unsuccessful");
       }
}
```

The printProperties( ) method performs the following operations:

1. Creates a statement, a local OracleResultSet, and a local ORDVideo object named vidObj, and populates vidObj with media data through the same process described in steps 1 through 5 of Example 2–23. In this method, you will be operating on the contents of the first row of the database table.
2. Gets the values of the properties in vidObj and prints them to the screen.
3. Handles any errors or exceptions raised by the code.

2.3.2.6 loadDataFromStream( ) Method

Example 2–26 shows a user-defined method named loadDataFromStream( ), which uses the *inter*Media loadDataFromInputStream( ) method to load media data into the application object.

Example 2–26 loadDataFromStream( ) Method (Video)

```
public void loadDataFromStream(OracleConnection con){
     try {
          [1] Statement s = con.createStatement( );
          OracleResultSet rs = (OracleResultSet) s.executeQuery
               ("select * from TVID where n = 2 for update ");
          int index = 0;
          while(rs.next( )){
               index = rs.getInt(1);
               OrdVideo vidObj = (OrdVideo) rs.getCustomDatum(2,
                    OrdVideo.getFactory( ));
               [2] FileInputStream  fStream  = new FileInputStream
                    ("testvid.dat");
               [3] vidObj.loadDataFromInputStream(fStream);
               [4] vidObj.getDataInFile("output2.dat");
               [5] fStream.close( );
               System.out.println("************AFTER getDataInFile ");
               [6] System.out.println("getContentLength output : " +
                    vidObj.getContentLength( ));
               [7] OraclePreparedStatement stmt1 = (OraclePreparedStatement)
                    con.prepareCall("update tvid set vid = ?
                    where n = " + index);
               stmt1.setCustomDatum(1,vidObj);
               stmt1.execute( );
               stmt1.close( ) ;
               index++;
          }
          System.out.println("load data from stream successful");
     }
     [8] catch(Exception e) {
          System.out.println("exception raised " + e);
          System.out.println("load data from stream unsuccessful");
     }
}
```

The loadDataFromStream( ) method performs the following operations:

1. Creates a statement, a local OracleResultSet, and a local ORDVideo object named vidObj, and populates vidObj with media data through the same process described in steps 1 through 5 of Example 2–23. In this method, you will be operating on the contents of the second row of the database table.
2. Creates a new FileInputStream object. This input stream contains the contents of the local file testvid.dat.
3. Uses the loadDataFromInputStream( ) method to load the media data in the input stream into the database ORDVideo object and into vidObj. This also sets the local field on vidObj, but not the database object.
4. Uses the getDataInFile( ) method to get the media data from the application ORDVideo object and load it into a file on the local system named output2.dat.
5. Closes the local input stream.
6. Gets the content length of vidObj and prints it to the screen to verify the success of the loading.
7. Creates and executes a SQL statement that will update the database ORDVideo object with the contents of vidObj. This update will set the attributes on the database object to match the application object.
8. Handles any errors or exceptions raised by the code.

2.3.2.7 otherMethods( ) Method

Example 2–27 shows a user-defined method named otherMethods( ), which attempts to use the processSourceCommand( ) method.

Example 2–27 otherMethods( ) Method (Video)

```
public void otherMethods(OracleConnection con){
     byte[ ] ctx[ ] = {new byte[4000]};
     byte[ ] res[ ] = {new byte[20]};
     [1] int suc = 1;
     try {
          [2] Statement s1 = con.createStatement( );
          OracleResultSet rs1 = (OracleResultSet)
          s1.executeQuery("select * from TVID where n = 1 for
               update ");
          int index1 = 0;
          while(rs1.next( )) {
               index1 = rs1.getInt(1);
               OrdVideo vidObj = (OrdVideo) rs1.getCustomDatum(2,
                    OrdVideo.getFactory( ));
```

```
                [3] try {
                     byte[ ] pSRes = vidObj.processSourceCommand(ctx,
                          "", "", res);
                     suc = 0;
                }
                [4] catch (Exception e) {
                     System.out.println("Expected Exception raised in
                          processSourceCommand(...)" );
                }
                [5] OraclePreparedStatement stmt1 = (OraclePreparedStatement)
                     con.prepareCall("update tvid set vid = ? where
                          n = " + index1);
                     stmt1.setCustomDatum(1,vidObj);
                     stmt1.execute( );
                     stmt1.close( ) ;
                     index1++;
               }
               rs1.close( );
               s1.close( );
          }
          [6] catch(Exception e){
               System.out.println("Exception raised " );
          }
          [7] if(suc == 1)
               System.out.println("other methods successful");
          else
               System.out.println("other methods unsuccessful");
}
```

The otherMethods( ) method performs the following operations:

1. Creates an integer that will be used to indicate the success or failure of the method and sets it initially to 1 (for success).
2. Creates a statement, a local OracleResultSet, and a local ORDVideo object named vidObj, and populates vidObj with media data through the same process described in steps 1 through 5 of Example 2–23. In this method, you will be operating on the contents of the first row of the database table.
3. Tries to call processSourceCommand( ) with no value specified for the command to be called on the server side. This should raise an exception, which means the code following the processSourceCommand( ) call will not be run and the code in the catch loop will. If an exception is not raised, then the method has failed and the success indicator is set to 0 (for failure).

4. Prints the expected exception that was raised in step 3.
5. Creates and executes a SQL statement that will update the database ORDVideo object with the contents of vidObj.
6. Handles any unexpected errors or exceptions raised by the code.
7. Prints the appropriate message to the screen based on the success or failure of the method.

2.3.2.8 loadDataFromByteArray( ) Method

Example 2–28 shows a user-defined method named loadDataFromByteArray( ), which uses the *inter*Media loadDataFromByteArray( ) method to load media data into the application object.

Example 2–28 loadDataFromByteArray( ) Method (Video)

```
public void loadDataFromByteArray(OracleConnection con){
     try {
          [1] Statement s = con.createStatement( );
          OracleResultSet rs = (OracleResultSet) s.executeQuery
               ("select * from TVID where n = 3 for update ");
          int index = 0;
          while(rs.next( )){
               index = rs.getInt(1);
               OrdVideo vidObj = (OrdVideo) rs.getCustomDatum(2,
                    OrdVideo.getFactory( ));
               [2] File ff = new File("testvid.dat");
               int fileLength = (int) ff.length( );
               byte[ ] data = new byte[fileLength];
               [3] FileInputStream  fStream  = new
                    FileInputStream("testvid.dat");
               [4] fStream.read(data,0,fileLength);
               [5] vidObj.loadDataFromByteArray(data);
               [6] fStream.close( );
               [7] vidObj.getDataInFile("output3.dat");
               [8] byte[ ] resArr = vidObj.getDataInByteArray( );
               [9] System.out.println("byte array length : " +
                    resArr.length);
               [10] FileOutputStream outStream = new FileOutputStream
                    ("output4.dat");
               [11] outStream.write(resArr);
               [12] outStream.close( );
               [13] InputStream inpStream = vidObj.getDataInStream( );
               int length = 32300;
```

```
                byte[ ] tempBuffer = new byte[32300];
                [14] int numRead = inpStream.read(tempBuffer,0,length);
                try {
                     [15] outStream = new FileOutputStream("output5.dat");
                     [16] while(numRead != -1) {
                          [17] if (numRead < 32300) {
                               length = numRead;
                               outStream.write(tempBuffer,0,length);
                               break;
                          }
                          [18] else
                               outStream.write(tempBuffer,0,length);
                          [19] numRead = inpStream.read(tempBuffer,0,length);
                     }
                }
                [20] finally {
                     outStream.close( );
                     inpStream.close( );
                }
                System.out.println("************AFTER getDataInFile ");
                [21] System.out.println(" getContentLength output : " +
                     vidObj.getContentLength( ));
                [22] OraclePreparedStatement stmt1 = (OraclePreparedStatement)
                     con.prepareCall("update tvid set vid = ? where
                     n = " + index);
                stmt1.setCustomDatum(1,vidObj);
                stmt1.execute( );
                stmt1.close( ) ;
                index++;
           }
      }
      [23] catch(Exception e) {
           System.out.println("exception raised " + e);
           System.out.println("loadData from byte array unsuccessful");
      }
}
```

The loadDataFromByteArray( ) method performs the following operations:

1. Creates a statement, a local OracleResultSet, and a local ORDVideo object named vidObj, and populates vidObj with media data through the same process described in steps 1 through 5 of Example 2–23. In this method, you will be operating on the contents of the third row of the database table.

2. Determines the size (in bytes) of the local file testvid.dat and creates a byte array of the same size.
3. Creates a new FileInputStream object. This input stream contains the contents of testvid.dat.
4. Reads the contents of the input stream into the byte array.
5. Uses the loadDataFromByteArray( ) method to load the media data in the byte array into the database ORDVideo object and into vidObj. This also sets the local field on vidObj, but not the database object.
6. Closes the input stream.
7. Uses the getDataInFile( ) method to get the media data from the application ORDVideo object and load it into a file on the local system named output3.dat.
8. Uses the getDataInByteArray( ) method to get the media data from the application ORDVideo object and load it into a local byte array named resArr.
9. Gets the length of resArr and prints it to the screen to verify the success of the loading.
10. Creates a new FileOutputStream object named outStream. This output stream will write data to a local file named output4.dat.
11. Writes the contents of resArr to output4.dat.
12. Closes the output stream.
13. Creates a new input stream named inpStream. Uses the getDataInStream( ) method to get the media data from the application ORDVideo object and store it in inpStream.
14. Reads 32300 bytes from the beginning (that is, at an offset of 0) of inpStream into the byte array tempBuffer. The integer numRead will be set to the total number of bytes read, or -1 if the end of the input stream has been reached. In this case, if loading is successful, numRead should be equal to 32300.
15. Re-opens OutStream. In this case, it will write data to a local file named output5.dat.
16. Runs the operations in the while loop if numRead is not equal to -1. The program should enter this loop.
17. Writes the number of bytes read into tempBuffer into outStream if numRead is less than 32300 (that is, if not all the data was read).
18. If numRead is 32300, writes 32300 bytes into outStream.

19. Attempts to read more data from the input stream into the byte array. If 32300 bytes of data are successfully read, then numRead will be set to -1 and the program will exit the loop. If there is still unread data in the input stream, then it will be read into the byte array and steps 17 and 18 will be repeated.
20. Closes both the input stream and the output stream after exiting the while loop.
21. Gets the content length of vidObj and prints it to the screen to verify the success of the loading.
22. Creates and executes a SQL statement that will update the database ORDVideo object with the contents of vidObj. This update will set the attributes on the database object to match the application object.
23. Handles any errors or exceptions raised by the code.

3

ORDAudio Reference Information

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

Methods invoked at the ORDAudio level that are handed off for processing to the database source plug-in or database format plug-in have `byte[ ] ctx[ ]` as a context parameter. In cases where a client system is connecting to a database server, the space for the parameter is created by the client (in the reference examples, 4000 bytes of space), but the content of the context parameter is generated by the server. The context parameter is passed from the client to the server for the processing of context information.

> **Note:** In the current release, not all source plug-ins or format plug-ins will use or generate the context parameter, but if you include the parameter as previously described, your application should work with any current or future source plug-ins or format plug-ins.

3.1 Prerequisites

You will need to include the following import statements in your Java file in order to run *inter*Media methods:

```
import java.sql.*;
import java.io.*;
import oracle.jdbc.driver.*;
import oracle.sql.*;
import oracle.ord.im.*;
```

The examples in this reference chapter are based on the assumption that the following operations have already been performed:

- A connection has been made to a table that contains a column of type ORDAudio.
- A local ORDAudio object named audObj has been created and populated with data.

For examples of making a connection and populating a local object, see Section 2.1.2.

3.2 Reference Information

This section presents reference information on the methods that operate on ORDAudio objects.

checkProperties( )

Format

public boolean checkProperties(byte[ ] ctx[ ])

Description

Checks if the properties stored in the media data of the local object are consistent with the attributes of the local object.

Parameters

ctx[ ]
The format plug-in context information. It is set to NULL if there is no context information.

Return Value

This method returns TRUE if the attribute values stored in the object attributes are the same as the properties stored in the BLOB data; FALSE otherwise.

Exceptions

java.sql.SQLException

Example

```
byte[ ] ctx[ ] = new byte[4000][1];
if(audObj.checkProperties(ctx))
     System.out.println("checkProperties successful");
```

where:

- ctx: contains the format plug-in context information.

clearLocal( )

Format

public void clearLocal( )

Description

Clears the source local field of the application ORDAudio object.

Parameters

None.

Return Value

None.

Exceptions

java.sql.SQLException

Example

```
audObj.clearLocal( )
```

closeSource( )

Format

public int closeSource(byte[ ] ctx[ ])

Description

Closes the ORDAudio file source.

Parameters

ctx[ ]
The source plug-in context information. It is set to NULL if there is no context information.

Return Value

This method returns 0 if the operation is successful, or an integer greater than 0 in case of failure.

Exceptions

java.sql.SQLException

Example

```
byte[ ] ctx[ ] = new byte[4000][1];
int i = audObj.closeSource(ctx);
if(i == 0)
     System.out.println("Source close successful");
else
     System.out.println("Source close unsuccessful");
```

where:

- ctx: contains the source plug-in context information.

deleteContent( )

Format

public void deleteContent( )

Description

Deletes the media data in the BLOB in the application ORDAudio object.

Parameters

None.

Return Value

None.

Exceptions

java.sql.SQLException

Example

```
audObj.deleteContent( );
```

export( )

Format

public void export (byte[ ] ctx[ ], String sourceType, String sourceLocation, String sourceName)

Description

Exports the data from the application ORDAudio object BLOB to the location specified in the parameters. The location is of the form:

sourceType: / / sourceLocation / sourceName

This method will only work if you are running Oracle8*i* database server version 8.1.7 or later.

See Chapter 4 of *Oracle8i interMedia Audio, Image, and Video User's Guide and Reference* for more information.

Parameters

ctx[ ]
The source plug-in context information. It is set to NULL if there is no context information.

sourceType
The source type to which the content will be exported. Only FILE is natively supported.

sourceLocation
The location on the database server to which the content will be exported.

sourceName
The name of the source to which the content will be exported.

Return Value

None.

Exceptions

java.sql.SQLException

Example

```
byte[ ] ctx[ ] = new byte[4000][1];
audObj.export(ctx,"FILE","AUDIODIR","complete.wav");
```

where:

- ctx: contains the source plug-in context information.
- FILE: is the source type to which the content will be exported.
- AUDIODIR: is the location to which the content will be exported.
- complete.wav: is the file to which the content will be exported.

getAllAttributes( )

Format

public CLOB getAllAttributes(byte[ ] ctx[ ])

Description

Gets all the attributes from the application ORDAudio object and puts them in a CLOB.

Parameters

ctx[ ]
The format plug-in context information. It is set to NULL if there is no context information.

Return Value

This method returns a CLOB that contains the attribute values of the ORDAudio object.

Exceptions

java.sql.SQLException

Example

```
byte[ ] ctx[ ] = new byte[4000][1];
CLOB attributes = audObj.getAllAttributes(ctx);
```

where:

- ctx: contains the format plug-in context information.

getAttribute( )

Format

public String getAttribute(byte[ ] ctx[ ], String name)

Description

Gets the value of a specified attribute from the application ORDAudio object as a String.

Parameters

ctx[ ]
The format plug-in context information. It is set to NULL if there is no context information.

name
The name of the attribute.

Return Value

This method returns the value of the specified attribute, as a String.

Exceptions

java.sql.SQLException

Example

```
byte[ ] ctx[ ] = new byte[4000][1];
int attribute = audObj.getAttribute(ctx,"numberOfChannels")
```

where:

- ctx: contains the format plug-in context information.
- numberOfChannels: is the attribute to get from the object.

getAudioDuration( )

Format

public int getAudioDuration( )

Description

Gets the audio duration of the application ORDAudio object.

Parameters

None.

Return Value

This method returns the audio duration of the ORDAudio object, in seconds.

Exceptions

java.sql.SQLException

Example

```
int audioDuration = audObj.getAudioDuration( );
```

getBFILE( )

Format

public oracle.sql.BFILE getBFILE( )

Description

Gets the BFILE attribute of the application ORDAudio object.

Parameters

None.

Return Value

This method returns the BFILE.

Exceptions

java.sql.SQLException

Example

```
BFILE audioBFILE = audObj.getBFILE( );
```

getComments( )

Format

public oracle.sql.CLOB getComments( )

Description

Gets the comments from the application ORDAudio object and puts them in a CLOB.

Parameters

None.

Return Value

This method returns a CLOB that contains the comments from the ORDAudio object.

Exceptions

java.sql.SQLException

Example

```
CLOB comments = audObj.getComments( )
```

getCompressionType( )

Format

public String getCompressionType( )

Description

Gets the compression type of the application ORDAudio object as a String.

Parameters

None.

Return Value

This method returns the compression type of the ORDAudio object, as a String.

Exceptions

java.sql.SQLException

Example

```
String compressionType = audObj.getComressionType( );
```

getContent( )

Format

public oracle.sql.BLOB getContent( )

Description

Gets the LOB locator from the application ORDAudio object.

Parameters

None.

Return Value

This method returns the LOB locator of the application ORDAudio object.

Exceptions

java.sql.SQLException

Example

```
BLOB localContent = audObj.getContent( );
```

getContentInLob( )

Format

public BLOB getContentInLob(byte[ ] ctx[ ], String mimetype[ ], String format[ ])

Description

Gets the content of the application ORDAudio object and puts it in a BLOB.

Parameters

ctx[ ]
The source plug-in context information. It is set to NULL if there is no context information.

mimetype[ ]
The MIME type of the content returned, stored in mimetype[0].

format[ ]
The format of the content returned, stored in format[0].

Return Value

This method returns the LOB content of the application ORDAudio object in a LOB locator.

Exceptions

java.sql.SQLException

Example

```
byte[ ] ctx[ ] = new byte[4000][1];
String mimeType[ ] = new String[1];
mimeType[0] = "video/x-msvideo"
String format[ ] = new String[1];
format[0] = "RIFF"
BLOB localContent = audObj.getContentInLob(ctx,mimeType,format);
```

where:

- ctx: contains the source plug-in context information.

- mimeType: is an array of Strings whose first value contains the MIME type.
- format: is an array of Strings whose first value contains the format.

getContentLength( )

Format

public int getContentLength( )

Description

Gets the content length of the media data in the application ORDAudio object.

Parameters

None.

Return Value

This method returns the content length of the media data.

Exceptions

java.sql.SQLException

Example

```
int contentLength = audObj.getContentLength( );
```

getContentLength(byte[ ][ ])

Format

public int getContentLength(byte[ ] ctx[ ])

Description

Gets the content length of the media data in the application ORDAudio object.

Parameters

ctx[ ]
The source plug-in context information. It is set to NULL if there is no context information.

Return Value

This method returns the content length of the media data.

Exceptions

java.sql.SQLException

Example

```
byte[ ] ctx[ ] = new byte[4000][1];
int contentLength = audObj.getContentLength(ctx);
```

where:

- ctx: contains the source plug-in context information.

getDataInByteArray( )

Format

public byte[ ] getDataInByteArray( )

Description

Gets data from the LOB locator of the application ORDAudio object and puts it in a local byte array.

Parameters

None.

Return Value

This method returns the byte array from which the data will be read.

Exceptions

java.sql.SQLException

java.io.IOException

java.lang.OutOfMemoryError

Example

```
byte[ ] byteArr = audObj.getDataInByteArray( );
```

getDataInFile( )

Format

public boolean getDataInFile(String filename)

Description

Gets data from the LOB locator of the application ORDAudio object and puts it in a local file.

Parameters

filename
The name of the file into which the data will be loaded.

Return Value

This method returns TRUE if loading is successful; FALSE otherwise.

Exceptions

java.sql.SQLException

java.io.IOException

Example

```
boolean load = audObj.getDataInFile("output1.dat");
if(load)
     System.out.println("getDataInFile completed successfully");
else
     System.out.println("Error in getDataInFile");
```

where:

- output1.dat: is the file into which the data will be loaded.

getDataInStream( )

Format

public InputStream getDataInStream( )

Description

Gets data from the LOB locator of the application ORDAudio object and puts it in a local input stream.

Parameters

None.

Return Value

This method returns the input stream from which the data will be read.

Exceptions

java.sql.SQLException

Example

```
InputStream inpStream = audObj.getDataInStream( );
```

getDescription( )

Format

public String getDescription( )

Description

Gets the description attribute of the application ORDAudio object.

Parameters

None.

Return Value

This method returns the description attribute as a String.

Exceptions

java.sql.SQLException

Example

```
String desc = audObj.getDescription( );
```

getEncoding( )

Format

public String getEncoding( )

Description

Gets the encoding of the application ORDAudio object as a String.

Parameters

None.

Return Value

This method returns the encoding of the ORDAudio object as a String.

Exceptions

java.sql.SQLException

Example

```
String encoding = audObj.getEncoding( );
```

getFormat( )

Format

public String getFormat( )

Description

Gets the format attribute of the application ORDAudio object as a String.

Parameters

None.

Return Value

This method returns the format attribute as a String.

Exceptions

java.sql.SQLException

Example

```
String format = audObj.getFormat( );
```

getMimeType( )

Format

public String getMimeType( )

Description

Gets the MIME type of the application ORDAudio object as a String.

Parameters

None.

Return Value

This method returns the MIME type of the ORDAudio object as a String.

Exceptions

java.sql.SQLException

Example

```
String mimeType = audObj.getMimeType( );
```

getNumberOfChannels( )

Format

public int getNumberOfChannels( )

Description

Gets the number of channels in the application ORDAudio object.

Parameters

None.

Return Value

This method returns the number of channels in the ORDAudio object as an integer.

Exceptions

java.sql.SQLException

Example

```
int channels = audObj.getNumberOfChannels( );
```

getSampleSize( )

Format

public int getSampleSize( )

Description

Gets the sample size of the application ORDAudio object as an integer.

Parameters

None.

Return Value

This method returns the sample size of the ORDAudio object.

Exceptions

java.sql.SQLException

Example

```
int sampleSize = audObj.getSampleSize( );
```

getSamplingRate( )

Format

public int getSamplingRate( )

Description

Gets the sampling rate of the application ORDAudio object as an integer.

Parameters

None.

Return Value

This method returns the sampling rate of the ORDAudio object in bytes per second.

Exceptions

java.sql.SQLException

Example

```
int samplingRate = audObj.getSamplingRate( );
```

getSource( )

Format

public String getSource( )

Description

Gets the object source information of the application ORDAudio object, including the source location, name, and type.

Parameters

None.

Return Value

This method returns a String containing the object source information.

Exceptions

java.sql.SQLException

Example

```
String source = audObj.getSource( );
```

getSourceLocation( )

Format

public String getSourceLocation( )

Description

Gets the source location of the application ORDAudio object as a String.

Parameters

None.

Return Value

This method returns a String containing the object source location.

Exceptions

java.sql.SQLException

Example

```
String location = audObj.getSourceLocation( );
```

getSourceName( )

Format

public String getSourceName( )

Description

Gets the source name of the application ORDAudio object as a String.

Parameters

None.

Return Value

This method returns a String containing the object source name.

Exceptions

java.sql.SQLException

Example

```
String name = audObj.getSourceName( );
```

getSourceType( )

Format

public String getSourceType( )

Description

Gets the source location of the application ORDAudio object as a String.

Parameters

None.

Return Value

This method returns a String containing the object source type.

Exceptions

java.sql.SQLException

Example

```
String type = audObj.getSourceType( );
```

getUpdateTime( )

Format

public java.sql.Timestamp getUpdateTime( )

Description

Gets a Timestamp object that contains information on when the application ORDAudio object was most recently updated.

Parameters

None.

Return Value

This method returns a Timestamp object that contains the time of the most recent update.

Exceptions

java.sql.SQLException

Example

```
Timestamp time = audObj.getUpdateTime( );
```

importData( )

Format

public void importData(byte[ ] ctx[ ])

Description

Imports data from an external source into the application ORDAudio object.

The srcType, srcLocation, and srcName attributes must all be defined for this method to work.

Parameters

ctx[ ]
The source plug-in context information. It is set to NULL if there is no context information.

Return Value

None.

Exceptions

java.sql.SQLException

Example

```
byte[ ] ctx[ ] = new byte[4000][1];
audObj.importData(ctx);
```

where:

- ctx: contains the source plug-in information.

importFrom( )

Format

public void importFrom(byte[ ] ctx[ ], String sourceType, String sourceLocation, String sourceName)

Description

Imports data from an external source into the application ORDAudio object. The location of the external source is of the form:

sourceType: / / sourceLocation / sourceName

Parameters

ctx[ ]
The source plug-in context information. See Chapter 4 of *Oracle8i interMedia Audio, Image, and Video User's Guide and Reference* for more information.

sourceType
The source type from which the data will be imported.

sourceLocation
The source location on the database server from which the data will be imported.

sourceName
The source name from which the data will be imported.

Return Value

None.

Exceptions

java.sql.SQLException

Example

```
byte[ ] ctx[ ] = new byte[4000][1];
audObj.importFrom("FILE","AUDIODIR","testaud.dat");
```

where:

- ctx: contains the source plug-in context information.
- FILE: is the type of the source from which the data will be imported.
- AUDIODIR: is the location of the file on the database server from which the data will be imported.
- testaud.dat: is the file from which the data will be imported.

isLocal( )

Format

public boolean isLocal( )

Description

Checks if the application ORDAudio object local attribute is set.

Parameters

None.

Return Value

This method returnsTRUE if the ORDAudio object local attribute is set; FALSE otherwise.

Exceptions

java.sql.SQLException

Example

```
if(audObj.isLocal( ))
     System.out.println("local attribute is set to true");
else
     System.out.println("lcoal attribute is set to false");
```

loadDataFromByteArray( )

Format

public boolean loadDataFromByteArray(byte[ ] byteArr)

Description

Loads data from the local byte buffer into the database ORDAudio object LOB locator and into the application object. It also calls setLocal( ), which sets the local field of the application ORDAudio object, but not the database object, and setUpdateTime(null), which sets the updateTime attribute to the SYSDATE of the database server.

Parameters

byteArr
The name of the local byte array from which the data will be loaded.

Return Value

This method returns TRUE if loading is successful; FALSE otherwise.

Exceptions

java.sql.SQLException

java.io.IOException

Example

```
byte[ ] data = new byte[32000];
FileInputStream fStream = new FileInputStream("testaud.dat");
fStream.read(data,0,32300);
boolean success = audObj.loadDataFromByteArray(data);
if(success)
     System.out.println("loadDataFromByteArray was successful");
else
     System.out.println("loadDataFromByteArray was unsuccessful");
```

where:

- data: is the local byte array from which the data will be loaded.

- testaud.dat: is a local file that contains 32,300 bytes of data.

loadDataFromFile( )

Format

public boolean loadDataFromFile(String filename)

Description

Loads data from the local file into the database ORDAudio object LOB locator and into the application object. It also calls setLocal( ), which sets the local field of the application ORDAudio object, but not the database object, and setUpdateTime(null), which sets the updateTime attribute to the SYSDATE of the database server.

Parameters

filename
The name of the local file from which to load data.

Return Value

This method returns TRUE if loading is successful; FALSE otherwise.

Exceptions

java.sql.SQLException

java.io.IOException

Example

```
audObj.loadDataFromFile("testaud.dat");
```

where:

- testaud.dat: is a local file that contains audio data.

loadDataFromInputStream( )

Format

public boolean loadDataFromInputStream(InputStream inpStream)

Description

Loads data from the local input stream into the database ORDAudio object LOB locator and into the application object. It also calls setLocal( ), which sets the local field of the application ORDAudio object, but not the database object, and setUpdateTime(null), which sets the updateTime attribute to the SYSDATE of the database server.

Parameters

inpStream
The name of the local input stream from which to load data.

Return Value

This method returns TRUE if loading is successful; FALSE otherwise.

Exceptions

java.sql.SQLException

java.io.IOException

Example

```
FileInputStream fStream = new FileInputStream("testaud.dat");
audObj.loadDataFromInputStream(fStream);
```

where:

- testaud.dat: is a local file that contains audio data.
- fStream: is the local input stream that will load audio data into the ORDAudio object.

openSource( )

Format

public int openSource(byte[ ] userarg, byte[ ] ctx[ ])

Description

Opens the ORDAudio file source.

Parameters

userarg
Permission-related parameters that are supplied by the user, such as READONLY. These may be used by user-defined source plug-ins.

ctx[ ]
The source plug-in context information. See Chapter 4 of *Oracle8i interMedia Audio, Image, and Video User's Guide and Reference* for more information.

Return Value

This method returns 0 if the operation is successful, or an integer greater than 0 in case of failure.

Exceptions

java.lang.Exception

Example

```
byte[ ] userarg = new byte[4000];
byte[ ] ctx[ ] = new byte[4000][1];
int i = audObj.openSource(userarg,ctx);
if(i == 0)
     System.out.println("openSource successful");
else
     System.out.println("openSource unsuccessful");
```

where:

- userarg: contains permission-related parameters.

- ctx: contains the source plug-in context information.

OrdAudio( )

Format

public OrdAudio( )

Description

Creates an ORDAudio object.

This method is the default ORDAudio constructor.

Parameters

None.

Return Value

None.

Exceptions

None.

Example

None.

processAudioCommand( )

Format

public byte[ ] processAudioCommand(byte[ ] ctx[ ], String command, String args, byte[ ] result[ ])

Description

Processes the specified command on the application ORDAudio object by calling the database processAudioCommand( ) method.

Parameters

ctx[ ]
The format plug-in context information. It is set to NULL if there is no context information.

command
The command to be executed. The command must be recognized by the database format plug-in.

args
The arguments of the command.

result[ ]
The results of the command.

Return Value

This method returns the results of the execution of the command.

Exceptions

java.sql.SQLException

Example

```
byte[ ] ctx[ ] = new byte[4000][1]
String command;
String arguments;
byte[ ] result;
//assign a command value to command
```

```
//assign any arguments to args
byte[ ] commandResults = audObj.processAudioCommand(ctx,command,
     arguments,result);
```

where:

- ctx: contains the format plug-in information.
- command: is the command to be run.
- arguments: contains any arguments required by the command.
- result: is the results of the command.

processSourceCommand( )

Format

public byte[ ] processSourceCommand(byte[ ] ctx[ ], String command, String args, byte[ ] result[ ])

Description

Processes the specified command on the application ORDAudio object by calling the database processSourceCommand( ) method.

Parameters

ctx[ ]
The source plug-in context information. See Chapter 4 of *Oracle8i interMedia Audio, Image, and Video User's Guide and Reference* for more information.

command
The command to be executed. The command must be recognized by the database source plug-in.

args
The arguments of the command to be executed.

result[ ]
The results of the command.

Return Value

This method returns the results of executing the command.

Exceptions

java.sql.SQLException

Example

```
byte[ ] ctx[ ] = new byte[4000][1];
String command;
String arguments;
byte[ ] result;
//assign a command value to command
```

```
//assign any arguments to args
byte[ ] commandResults = audObj.processSourceCommand(ctx,command,
     arguments,result);
```

where:

- ctx: contains the source plug-in information.
- command: is the command to be run.
- arguments: contains any arguments required by the command.
- result: is the results of the command.

readFromSource( )

Format

public int readFromSource(byte[ ] ctx[ ], int startpos, int numbytes, byte[ ] buffer)

Description

Reads data from the comments field of the application ORDAudio object.

Parameters

ctx[ ]
The source plug-in context information. See Chapter 4 of *Oracle8i interMedia Audio, Image, and Video User's Guide and Reference* for more information.

startpos
The initial position in the comments field.

numbytes
The number of bytes to be read.

buffer
The buffer into which to read the content.

Return Value

This method returns the number of bytes read.

Exceptions

java.sql.SQLException

Example

```
byte[ ] ctx[ ] = new byte[4000][1];
byte[ ] commentBuffer = new byte[12];
int i = audObj.readFromSource(ctx,0,12,commentBuffer);
```

where:

- ctx: contains the source plug-in context information.

- 0: is the position to begin reading from the comments field.
- 12: is the number of bytes to be read.
- commentBuffer: is the location to which the data will be read.

setAudioDuration( )

Format

public void setAudioDuration(int audioDuration)

Description

Sets the audio duration of the application ORDAudio object.

setProperties( ) will automatically set this attribute; use this method only if you are not using setProperties( ). Also, this method will set only the attribute value; it does not change the media file itself.

Parameters

audioDuration
The audio duration (in seconds) to be set in the ORDAudio object.

Return Value

None.

Exceptions

java.sql.SQLException

Example

```
audObj.setAudioDuration(16);
```

where:

- 16: is the audio duration to be set, in seconds.

setComments( )

Format

public void setComments(oracle.sql.CLOB comments)

Description

Sets the comments in the application ORDAudio object.

The comments attribute is reserved for use by *inter*Media. You can set your own value, but it could be overwritten by *inter*Media Annotator.

Parameters

comments
A CLOB that contains comments for the ORDAudio object.

Return Value

None.

Exceptions

java.sql.SQLException

Example

```
audObj.setComments(commentsData);
```

where:

- commentsData: is a CLOB that contains comments to be set.

setCompressionType( )

Format

public void setCompressionType(String CompressionType)

Description

Sets the compression type of the application ORDAudio object.

setProperties( ) will automatically set this attribute; use this method only if you are not using setProperties( ). Also, this method will set only the attribute value; it does not change the media file itself.

Parameters

CompressionType
The compression type of the ORDAudio object, as a String.

Return Value

None.

Exceptions

java.sql.SQLException

Example

```
audObj.setCompressionType("8BITMONOAUDIO");
```

where:

- 8BITMONOAUDIO: is the compression type.

setDescription( )

Format

public void setDescription(String description)

Description

Sets the description attribute of the application ORDAudio object.

Parameters

description
A String that describes the contents of the ORDAudio object.

Return Value

None.

Exceptions

java.sql.SQLException

Example

```
audObj.setDescription("My audio file");
```

where:

- My audio file: is the description to be set in the object.

setEncoding( )

Format

public void setEncoding(String encoding)

Description

Sets the encoding of the application ORDAudio object.

setProperties( ) will automatically set this attribute; use this method only if you are not using setProperties( ). Also, this method will set only the attribute value; it does not change the media file itself.

Parameters

encoding
The encoding of the contents of an ORDAudio object, as a String.

Return Value

None.

Exceptions

java.sql.SQLException

Example

```
audObj.setEncoding("MULAW");
```

where:

- MULAW: is the encoding to be set.

setFormat( )

Format

public void setFormat(String format)

Description

Sets the format attribute of the application ORDAudio object.

setProperties( ) will automatically set this attribute; use this method only if you are not using setProperties( ). Also, this method will set only the attribute value; it does not change the media file itself.

Parameters

format
The format of the contents of the ORDAudio object, as a String.

Return Value

None.

Exceptions

java.sql.SQLException

Example

```
audObj.setFormat("AUFF");
```

where:

- AUFF: is the format to be set.

setKnownAttributes( )

Format

```
public void setKnownAttributes(String knownformat, String knownencoding,
                               int knownnumberofchannels, int knownsamplingrate,
                               int knownsamplesize, String knowncompressiontype,
                           int knownaudioduration)
```

Description

Sets the known attributes of the ORDAudio object.

setProperties( ) will automatically set these attributes; use this method only if you are not using setProperties( ). Also, this method will set only the attribute values; it does not change the media file itself.

Parameters

knownformat
The audio data format.

knownencoding
The audio data encoding.

knownnumberofchannels
The number of channels.

knownsamplingrate
The sampling rate.

knownsamplesize
The sample size.

knowncompressiontype
The compression type.

knownaudioduration
The audio duration.

Return Value

None.

Exceptions

java.sql.SQLException

Example

```
audObj.setKnownAttributes("AUFF","MULAW",1,8,8,"8BITMONOAUDIO",16);
```

where:

- AUFF: is the format to be set.
- MULAW: is the encoding to be set.
- 1: is the number of channels to be set.
- 8: is the sampling rate to be set, in samples per second.
- 8: is the sample size to be set.
- 8BITMONOAUDIO: is the compression type to be set.
- 16: is the audio duration to be set, in seconds.

setLocal( )

Format

public void setLocal( )

Description

Sets the source local field of the application ORDAudio object.

Parameters

None.

Return Value

None.

Exceptions

java.sql.SQLException

Example

```
audObj.setLocal( );
```

setMimeType( )

Format

public void setMimeType(String MimeType)

Description

Sets the MIME type of the application ORDAudio object.

setProperties( ) will automatically set this attribute; use this method only if you are not using setProperties( ). Also, this method will set only the attribute value; it does not change the media file itself.

Parameters

MimeType
The MIME type of the contents of the ORDAudio object, as a String.

Return Value

None.

Exceptions

java.sql.SQLException

Example

```
audObj.setMimeType("audio/basic");
```

where:

- audio/basic: is the MIME type to be set.

setNumberOfChannels( )

Format

public void setNumberOfChannels(int numberOfChannels)

Description

Sets the number of the channels in the application ORDAudio object.

setProperties( ) will automatically set this attribute; use this method only if you are not using setProperties( ). Also, this method will set only the attribute value; it does not change the media file itself.

Parameters

numberOfChannels
The number of channels to be set.

Return Value

None.

Exceptions

java.sql.SQLException

Example

```
audObj.setNumberOfChannels(1);
```

where:

- 1: is the number of channels to be set.

setProperties(byte[ ][ ])

Format

public void setProperties(byte[ ] ctx[ ])

Description

Reads the audio data, extracts the properties, and sets the properties in the application ORDAudio object.

The properties to be set include format, encoding, number of channels, sampling rate, sample size, compression type, and audio duration.

Parameters

ctx[ ]
The format plug-in context information. It is set to NULL if there is no context information.

Return Value

None.

Exceptions

java.sql.SQLException

Example

```
byte[ ] ctx[ ] = new byte[4000][1];
audObj.setProperties(ctx);
```

where:

- ctx: contains the format plug-in context information.

setProperties(byte[ ][ ], boolean)

Format

public void setProperties(byte[ ] ctx[ ], boolean setComments)

Description

Reads the media data, extracts the properties, and sets the properties in the application ORDAudio object.

The properties to be set include format, encoding, number of channels, sampling rate, sample size, compression type, and audio duration.

Parameters

ctx[ ]
The format plug-in context information. It is set to NULL if there is no context information.

setComments
A Boolean value to determine whether or not to set the comments in the ORDAudio object. If TRUE, the comments field is populated with a set of format and application properties of the audio object in XML. If FALSE, the comments field remains unpopulated. The default value is FALSE.

Return Value

None.

Exceptions

java.sql.SQLException

Example

```
byte[ ] ctx[ ] = new byte[4000][1];
audObj.setProperties(ctx,true);
```

where:

- ctx: contains the format plug-in context information.

- true: indicates that the comments field will be populated.

setSampleSize( )

Format

public void setSampleSize(int sampleSize)

Description

Sets the sample size in the application ORDAudio object.

setProperties( ) will automatically set this attribute; use this method only if you are not using setProperties( ). Also, this method will set only the attribute value; it does not change the media file itself.

Parameters

sampleSize
The sample size of the ORDAudio object.

Return Value

None.

Exceptions

java.sql.SQLException

Example

```
audObj.setSampleSize(8);
```

where:

- 8: is the sample size to be set.

setSamplingRate( )

Format

public void setSamplingRate(int samplingRate)

Description

Sets the sampling rate of the application ORDAudio object.

setProperties( ) will automatically set this attribute; use this method only if you are not using setProperties( ). Also, this method will set only the attribute value; it does not change the media file itself.

Parameters

samplingRate
The sampling rate value to be set, in samples per second.

Return Value

None.

Exceptions

java.sql.SQLException

Example

```
audObj.setSamplingRate(8);
```

where:

- 8: is the sampling rate to be set, in samples per second.

setSource( )

Format

public void setSource(String sourceType, String sourceLocation, String sourceName)

Description

Sets the application ORDAudio object source information.

Parameters

sourceType
The type of the source.

sourceLocation
The location of the source.

sourceName
The name of the source.

Return Value

None.

Exceptions

java.sql.SQLException

Example

```
audObj.setSource("FILE","AUDIODIR","audio.au");
```

where:

- FILE: is the source type.
- AUDIODIR: is the source location.
- audio.au: is the source name.

setUpdateTime( )

Format

public void setUpdateTime(java.sql.Timestamp currentTime)

Description

Sets the update time in the application ORDAudio object to the current time.

setProperties( ) will automatically set this attribute; use this method only if you are not using setProperties( ). Also, this method will set only the attribute value; it does not change the media file itself.

Parameters

currentTime
The current time, which will be set in the ORDAudio object. This value should be NULL; the method will then use the SYSDATE of the database server.

Return Value

None.

Exceptions

java.sql.SQLException

Example

```
audObj.setUpdateTime(null);
```

trimSource( )

Format

public int trimSource(byte[ ] ctx[ ], int newLen)

Description

Trim the file source of the application ORDAudio object to the given length.

Parameters

ctx[ ]
The source plug-in context information. See Chapter 4 of *Oracle8i interMedia Audio, Image, and Video User's Guide and Reference* for more information.

newLen
The length to which the source will be trimmed.

Return Value

This method returns 0 if the operation is successful, or an integer greater than 0 in case of failure.

Exceptions

java.sql.SQLException

Example

```
byte[ ] ctx[ ] = new byte[4000][1];
int i = audObj.trimSource(ctx,10);
if (i == 0)
     System.out.println("trimSource successful");
else
     System.out.println("trimSource unsuccessful");
```

where:

- ctx: contains the source plug-in context information.
- 10: is the new length of the source.

writeToSource( )

Format

public int writeToSource(byte[ ] ctx[ ], int startpos, int numbytes, byte[ ] buffer)

Description

Writes data to the comments field of the application ORDAudio object.

Parameters

ctx[ ]
The source plug-in context information. See Chapter 4 of *Oracle8i interMedia Audio, Image, and Video User's Guide and Reference* for more information.

startpos
The initial position in the comments field.

numbytes
The number of bytes to be written.

buffer
The buffer containing the content to be written.

Return Value

This method returns the number of bytes written.

Exceptions

java.sql.SQLException

Example

```
byte[ ] ctx[ ] = new byte[4000][1];
byte[ ] data = new byte[20];
//populate data with 20 bytes of content
int i = audObj.writeToSource(ctx,1,20,data);
```

where:

- ctx: contains the source plug-in context information.

- 1: is the position in the comments field where writing will begin.
- 20: is the number of bytes to be written.
- data: contains the content to be written.

4

ORDImage Reference Information

*inter*Media Java Classes describes the ORDImage object type, which supports the storage and management of image data.

Methods invoked at the ORDImage level that are handed off for processing to the database source plug-in have `byte[ ] ctx[ ]` as a context parameter. In cases where a client system is connecting to a database server, the space for the parameter is created by the client (in the reference examples, 4000 bytes of space), but the content of the context parameter is generated by the server. The context parameter is passed from the client to the server for the processing of context information.

> **Note:** In the current release, not all source plug-ins will use or generate the context parameter, but if you include the parameter as previously described, your application should work with any current or future source plug-ins.

See *Oracle8i interMedia Audio, Image, and Video User's Guide and Reference* for more information.

4.1 Prerequisites

You will need to include the following import statements in your Java file in order to run *inter*Media methods:

```
import java.sql.*;
import java.io.*;
import oracle.jdbc.driver.*;
import oracle.sql.*;
import oracle.ord.im.*;
```

The examples in this reference chapter are based on the assumption that the following operations have already been performed:

- A connection has been made to a table that contains a column of type ORDImage.
- A local ORDImage object named imgObj has been created and populated with data.

For examples of making a connection and populating a local object, see Section 2.2.2.

4.2 Reference Information

This section presents reference information on the methods that operate on ORDImage objects.

checkProperties( )

Format

public boolean checkProperties( )

Description

Checks if the properties stored in the media data of the local object are consistent with the attributes of the local object.

Parameters

None.

Return Value

This method returns TRUE if the attribute values stored in the object attributes are the same as the properties stored in the image data; FALSE otherwise.

Exceptions

java.sql.SQLException

Example

```
if(imgObj.checkProperties( ))
     System.out.println("checkProperties successful");
```

clearLocal( )

Format

public void clearLocal( )

Description

Clears the source local field of the application ORDImage object.

Parameters

None.

Return Value

None.

Exceptions

java.sql.SQLException

Example

```
imgObj.clearLocal( );
```

copy( )

Format

public void copy(OrdImage dest)

Description

Copies image data from the application ORDImage object source to a destination ORDImage object.

Parameters

dest
The ORDImage object to which the data will be copied.

Return Value

None.

Exceptions

java.sql.SQLException

Example

```
//create and populate an object named imgObj2
imgObj.copy(imgObj2);
```

where

- imgObj2: is an ORDImage object that will receive the image data from imgObj.

deleteContent( )

Format

public void deleteContent( )

Description

Deletes the media data in the BLOB in the application ORDImage object.

Parameters

None.

Return Value

None.

Exceptions

java.sql.SQLException

Example

```
audObj.deleteContent( );
```

export( )

Format

public void export (byte[ ] ctx[ ], String sourceType, String sourceLocation, String sourceName)

Description

Exports the data from the application ORDImage object to the location specified in the parameters. The location is of the form:

sourceType: / / sourceLocation / sourceName

This method will work only if you are running Oracle database release 8.1.7 or later.

See Chapter 4 of *Oracle8i interMedia Audio, Image, and Video User's Guide and Reference* for more information.

Parameters

ctx[ ]
The source plug-in context information. It is set to NULL if there is no context information.

sourceType
The source type to which the content will be exported. Only "FILE" is natively supported.

sourceLocation
The location on the database server to which the content will be exported.

sourceName
The source name to which the content will be exported.

Return Value

None.

Exceptions

java.sql.SQLException

Example

```
byte[ ] ctx[ ] = new byte[4000][1];
imgObj.export(ctx,"FILE","IMAGEDIR","image.gif");
```

where:

- ctx: contains the source plug-in context information.
- FILE: is the source type to which the content will be exported.
- IMAGEDIR: is the location on the database server to which the content will be exported.
- image.gif: is the file to which the content will be exported.

getBFILE( )

Format

public oracle.sql.BFILE getBFILE( )

Description

Gets the BFILE attribute of the application ORDImage object.

Parameters

None.

Return Value

This method returns the BFILE.

Exceptions

java.sql.SQLException

Example

```
BFILE imageBFILE = imgObj.getBFILE( );
```

getCompressionFormat( )

Format

public String getCompressionFormat( )

Description

Gets the compression format attribute of the application ORDImage object as a String.

Parameters

None.

Return Value

This method returns the compression format attribute, as a String.

Exceptions

java.sql.SQLException

Example

```
String compression = imgObj.getCompressionFormat( );
```

getContent( )

Format

public oracle.sql.BLOB getContent( )

Description

Gets the LOB locator from the application ORDImage object.

Parameters

None.

Return Value

This method returns the LOB locator of the application object.

Exceptions

java.sql.SQLException

Example

```
BLOB localContent = imgObj.getContent( );
```

getContentFormat( )

Format

public String getContentFormat( )

Description

Gets the content format attribute of the application ORDImage object as a String.

Parameters

None.

Return Value

This method returns the content format attribute, as a String.

Exceptions

java.sql.SQLException

Example

```
String format = imgObj.getContentFormat( );
```

getContentLength( )

Format

public int getContentLength( )

Description

Gets the content length of the media data in the application ORDImage object.

Parameters

None.

Return Value

This method returns the content length of the media data, in bytes.

Exceptions

java.sql.SQLException

Example

```
int length = imgObj.getContentLength( );
```

getDataInByteArray( )

Format

public byte[ ] getDataInByteArray( )

Description

Gets data from the LOB locator of the application ORDImage object and puts it in a local byte array.

Parameters

None.

Return Value

This method returns the byte array from which the data will be read.

Exceptions

java.sql.SQLException

java.io.IOException

java.lang.OutOfMemoryError

Example

```
byte[ ] byteArr = imgObj.getDataInByteArray( );
```

getDataInFile( )

Format

public boolean getDataInFile(String filename)

Description

Gets data from the LOB locator of the application ORDImage object and puts it in a local file.

Parameters

filename
The file into which the data will be loaded.

Return Value

This method returns TRUE if loading is successful; FALSE otherwise.

Exceptions

java.sql.SQLException

java.io.IOException

Example

```
boolean load = imgObj.getDataInFile("output1.dat");
if(load)
     System.out.println("getDataInFile completed successfully");
else
     System.out.println("Error in getDataInFile");
```

where:

- output1.dat: is the file into which the data will be loaded.

getDataInStream( )

Format

public InputStream getDataInStream( )

Description

Gets data from the LOB locator of the application ORDImage file and puts it in a local input stream.

Parameters

None.

Return Value

This method returns the input stream from which the data will be read.

Exceptions

java.sql.SQLException

Example

```
InputStream inpStream = imgObj.getDataInStream( );
```

getFormat( )

Format

public String getFormat( )

Description

Gets the format attribute of the application ORDImage object as a String.

Parameters

None.

Return Value

This method returns the format attribute as a String.

Exceptions

java.sql.SQLException

Example

```
String format = imgObj.getFormat( );
```

getHeight( )

Format

public int getHeight( )

Description

Gets the height of the application ORDImage object.

Parameters

None.

Return Value

This method returns the height of the ORDImage object, in pixels.

Exceptions

java.sql.SQLException

Example

```
int height = imgObj.getHeight( );
```

getMimeType( )

Format

public String getMimeType( )

Description

Gets the MIME type of the application ORDImage object as a String.

Parameters

None.

Return Value

This method returns the MIME type of the ORDImage object, as a String.

Exceptions

java.sql.SQLException

Example

```
String mime = imgObj.getMimeType( );
```

getSource( )

Format

public String getSource( )

Description

Gets the application ORDImage object source information, including the source location, name, and type.

Parameters

None.

Return Value

This method returns a String containing the object source information.

Exceptions

java.sql.SQLException

Example

```
String sourceName = imgObj.getSource( );
```

getSourceLocation( )

Format

public String getSourceLocation( )

Description

Gets the application ORDImage object source location as a String.

Parameters

None.

Return Value

This method returns a String containing the object source location.

Exceptions

java.sql.SQLException

Example

```
String location = imgObj.getSourceLocation( );
```

getSourceName( )

Format

public String getSourceName( )

Description

Gets the application ORDImage object source name as a String.

Parameters

None.

Return Value

This method returns a String containing the object source name.

Exceptions

java.sql.SQLException

Example

```
String name = imgObj.getSourceName( );
```

getSourceType( )

Format

public String getSourceType( )

Description

Gets the application ORDImage object source location as a String.

Parameters

None.

Return Value

This method returns a String containing the object source type.

Exceptions

java.sql.SQLException

Example

```
String type = imgObj.getSourceType( );
```

getUpdateTime( )

Format

public java.sql.Timestamp getUpdateTime( )

Description

Gets a Timestamp object that contains information on when the application ORDImage object was most recently updated.

Parameters

None.

Return Value

This method returns a Timestamp object that contains the time of the most recent update.

Exceptions

java.sql.SQLException

Example

```
Timestamp time = imgObj.getUpdateTime( );
```

getWidth( )

Format

public int getWidth( )

Description

Gets the width of the application ORDImage object.

Parameters

None.

Return Value

This method returns the width of the ORDImage object, in pixels.

Exceptions

java.sql.SQLException

Example

```
int width = imgObj.getWidth( );
```

importData( )

Format

public void importData(byte[ ] ctx[ ])

Description

Imports data from an external source into the application ORDImage object.

The srcType, srcLocation, and srcName attributes must all be defined for this method to work.

Parameters

ctx[ ]
The source plug-in context information. It is set to NULL if there is no context information.

Return Value

None.

Exceptions

java.sql.SQLException

Example

```
byte[ ] ctx[  ] = new byte[4000][1];
imgObj.importData(ctx)
```

where:

- ctx: contains the source plug-in context information.

importFrom( )

Format

public void importFrom(byte[ ] ctx[ ], String sourceType, String sourceLocation, String sourceName)

Description

Imports data from an external source into the application ORDImage object. The location of the external source is of the form:

sourceType: / / sourceLocation / sourceName

Parameters

ctx[ ]
The source plug-in context information. See Chapter 4 of *Oracle8i interMedia Audio, Image, and Video User's Guide and Reference* for more information.

sourceType
The source type from which the data will be imported.

sourceLocation
The source location from which the data will be imported.

sourceName
The source name from which the data will be imported.

Return Value

None.

Exceptions

java.sql.SQLException

Example

```
byte[ ] ctx[  ] = new byte[4000][1];
imgObj.importFrom(ctx,"FILE","IMAGEDIR","testimg.dat");
```

where:

- ctx: contains the source plug-in context information.
- FILE: is the type of the source from which the data will be imported.
- IMAGEDIR: is the location of the file from which the data will be imported.
- testimg.dat: is the file from which the data will be imported.

isLocal( )

Format

public boolean isLocal( )

Description

Checks if the application ORDImage object local attribute is set.

Parameters

None.

Return Value

This method returns TRUE if the ORDImage object local attribute is set; FALSE otherwise.

Exceptions

java.sql.SQLException

Example

```
if(imgObj.isLocal( ))
     System.out.println("local attribute is true");
else
     System.out.println("local attribute is false");
```

loadDataFromByteArray( )

Format

public boolean loadDataFromByteArray(byte[ ] byteArr)

Description

Loads data from the local byte buffer into the database ORDImage object LOB locator and into the application object. It also calls setLocal( ), which sets the local field of the application ORDImage object, but not the database object, and setUpdateTime(null), which sets the updateTime attribute to the SYSDATE of the database server.

Parameters

byteArr
The name of the local byte array from which to load data.

Return Value

This method returns TRUE if loading is successful; FALSE otherwise.

Exceptions

java.sql.SQLException

java.io.IOException

Example

```
byte[ ] data = new byte[32300];
FileInputStream fStream = new FileInputStream("testimg.dat");
fStream.read(data,0,32300);
boolean success = imgObj.loadDataFromByteArray(data);
if(success)
     System.out.println("loadDataFromByteArray was successful");
else
     System.out.println("loadDataFromByteArray was unsuccessful");
```

where:

- data: is the local byte array from which the data will be loaded.

- testimg.dat: is a local file that contains 32,300 bytes of data.

loadDataFromFile( )

Format

public boolean loadDataFromFile(String filename)

Description

Loads data from the local file into the database ORDImage object LOB locator and into the application object. It also calls setLocal( ), which sets the local field of the application ORDImage object, but not the database object, and setUpdateTime(null), which sets the updateTime attribute to the SYSDATE of the database server.

Parameters

filename
The name of the local file from which to load data.

Return Value

This method returns TRUE if loading is successful; FALSE otherwise.

Exceptions

java.sql.SQLException

java.io.IOException

Example

```
imgObj.loadDataFromFile("testimg.dat");
```

where:

- testimg.dat: is a local file that contains image data.

loadDataFromInputStream( )

Format

public boolean loadDataFromInputStream(InputStream inpStream)

Description

Loads data from the local input stream into the database ORDImage object LOB locator and into the application object. It also calls setLocal( ), which sets the local field of the application ORDImage object, but not the database object, and setUpdateTime(null), which sets the updateTime attribute to the SYSDATE of the database server.

Parameters

inpStream
The name of the local input stream from which to load data.

Return Value

This method returns TRUE if loading is successful; FALSE otherwise.

Exceptions

java.sql.SQLException

java.io.IOException

Example

```
FileInputStream fStream = new FileInputStream("testimg.dat");
imgObj.loadDataFromInputStream(fStream);
```

where:

- testimg.dat: is a local file that contains image data.
- fStream: is the local input stream that will load image data into the ORDImage object.

OrdImage( )

Format

public OrdImage( )

Description

Creates an ORDImage object.

This method is the default ORDImage constructor.

Parameters

None.

Return Value

None.

Exceptions

None.

Example

None.

process( )

Format

public void process(String command)

Description

Executes a given command on the application ORDImage object.

For more information on the commands that can be processed, see *Oracle8i interMedia Audio, Image, and Video User's Guide and Reference*.

Parameters

command
The command to be executed.

Return Value

None.

Exceptions

java.sql.SQLException

Example

```
imgObj.process("fileFormat=JFIF");
```

where:

- fileFormat=JFIF: is the command to be executed.

processCopy( )

Format

public void processCopy(String command, OrdImage dest)

Description

Copies image data from the application ORDImage object to a destination ORDImage object and modifies the copy according to the specified command.

Parameters

command
The command to be executed.

dest
The object that will receive the results of the command.

Return Value

None.

Exceptions

java.sql.SQLException

Example

```
//create and populate an OrdImage object named imgObj2
imgObj.processCopy("maxScale=32 32, fileFormat= GIFF", imgObj2);
```

where:

- maxScale=32 32, fileFormat= GIFF: is the command to be executed.
- imgObj2: is the object that will receive the results of the command.

setCompressionFormat( )

Format

public void setCompressionFormat(String CompressionFormat)

Description

Sets the compression format attribute of the application ORDImage object.

setProperties( ) will automatically set this attribute; use this method only if you are not using setProperties( ). Also, this method will set only the attribute value; it does not change the media file itself.

Parameters

CompressionFormat
The compression format to be set.

Return Value

None.

Exceptions

java.sql.SQLException

Example

None.

setContentFormat( )

Format

public void setContentFormat(String ContentFormat)

Description

Sets the content format attribute of the application ORDImage object.

setProperties( ) will automatically set this attribute; use this method only if you are not using setProperties( ). Also, this method will set only the attribute value; it does not change the media file itself.

Parameters

ContentFormat
The content format to be set.

Return Value

None.

Exceptions

java.sql.SQLException

Example

None.

setContentLength( )

Format

public void setContentLength(int newContentLength)

Description

Sets the content length of the media data in the application ORDImage object.

setProperties( ) will automatically set this attribute; use this method only if you are not using setProperties( ). Also, this method will set only the attribute value; it does not change the media file itself.

Parameters

newContentLength
The new content length to be set, in bytes.

Return Value

None.

Exceptions

java.sql.SQLException

Example

None.

setFormat( )

Format

public void setFormat(String Format)

Description

Sets the format attribute of the application ORDImage object.

setProperties( ) will automatically set this attribute; use this method only if you are not using setProperties( ). Also, this method will set only the attribute value; it does not change the media file itself.

Parameters

Format
The format of the contents of the ORDImage object, as a String.

Return Value

None.

Exceptions

java.sql.SQLException

Example

None.

setHeight( )

Format

public void setHeight(int newHeight)

Description

Sets the height of the application ORDImage object.

setProperties( ) will automatically set this attribute; use this method only if you are not using setProperties( ). Also, this method will set only the attribute value; it does not change the media file itself.

Parameters

newHeight
The new height to be set, in pixels.

Return Value

None.

Exceptions

java.sql.SQLException

Example

```
imgObj.setHeight(24);
```

where:

- 24: is the height to be set, in pixels.

setLocal( )

Format

public void setLocal( )

Description

Sets the source local field of the application ORDImage object.

Parameters

None

Return Value

None.

Exceptions

java.sql.SQLException

Example

```
imgObj.setLocal( );
```

setMimeType( )

Format

public void setMimeType(String MimeType)

Description

Sets the MIME type of the application ORDImage object.

setProperties( ) will automatically set this attribute; use this method only if you are not using setProperties( ). Also, this method will set only the attribute value; it does not change the media file itself.

Parameters

MimeType
The MIME type of the contents of the ORDImage object, as a String.

Return Value

None.

Exceptions

java.sql.SQLException

Example

```
imgObj.setMimeType("image/bmp");
```

where:

- image/bmp: is the MIME type to be set.

setSource( )

Format

public void setSource(String sourceType, String sourceLocation, String sourceName)

Description

Sets the application ORDImage object source information.

Parameters

sourceType
The type of the source.

sourceLocation
The location of the source.

sourceName
The name of the source.

Return Value

None.

Exceptions

java.sql.SQLException

Example

```
imgObj.setSource("FILE","IMAGEDIR","jdoe.gif");
```

where:

- FILE: is the source type.
- IMAGEDIR: is the source location.
- jdoe.gif: is the source name.

setProperties( )

Format

public void setProperties( )

Description

Reads the image data, extracts attributes according to the values stored in the content data header, and sets the application ORDImage object attributes.

The properties to be set include height, width, data size of the on-disk image, file type, image type, compression type, and MIME type.

Parameters

None.

Return Value

None.

Exceptions

java.sql.SQLException

Example

```
imgObj.setProperties( );
```

setProperties(String)

Format

public void setProperties(String command)

Description

Sets the application ORDImage object attributes according to the values stored in the given String. This method is used for foreign image files. For more information on foreign image files and the properties that can be set, see *Oracle8i interMedia Audio, Image, and Video User's Guide and Reference*.

Parameters

command
The object attribute values to be set.

Return Value

None.

Exceptions

java.sql.SQLException

Example

```
String properties = "width=123 height=321 compressionformat=NONE
     userString=DJM dataOffset=128 scanlineOrder=INVERSE
     pixelOrder=REVERSE interleaving=BIL numberOfBands=1
     defaultChannelSelection=1";
imgObj.setProperties(properties);
```

where:

- properties: is a String that contains the properties to be set.

setUpdateTime( )

Format

public void setUpdateTime(java.sql.Timestamp currentTime)

Description

Sets the update time in the application ORDImage object to the current time.

setProperties( ) will automatically set this attribute; use this method only if you are not using setProperties( ). Also, this method will set only the attribute value; it does not change the media file itself.

Parameters

currentTime
The current time, which will be set in the ORDImage object. This value should be NULL; the method will then use the SYSDATE of the database server.

Return Value

None.

Exceptions

java.sql.SQLException

Example

```
imgObj.setUpdateTime(null);
```

setWidth( )

Format

public void setWidth(int newWidth)

Description

Sets the width of the application ORDImage object.

setProperties( ) will automatically set this attribute; use this method only if you are not using setProperties( ). Also, this method will set only the attribute value; it does not change the media file itself.

Parameters

newWidth
The width to be set, in pixels.

Return Value

None.

Exceptions

java.sql.SQLException

Example

```
imgObj.setWidth(24);
```

where:

- 24: is the width to be set, in pixels.

5

ORDVideo Reference Information

*inter*Media Java Classes describes the ORDVideo object type, which supports the storage and management of video data.

Methods invoked at the ORDVideo level that are handed off for processing to the database source plug-in or database format plug-in have `byte[ ] ctx[ ]` as a context parameter. In cases where a client system is connecting to a database server, the space for the parameter is created by the client (in the reference examples, 4000 bytes of space), but the content of the context parameter is generated by the server. The context parameter is passed from the client to the server for the processing of context information.

> **Note:** In the current release, not all source plug-ins or format plug-ins will use or generate the context parameter, but if you include the parameter as previously described, your application should work with any current or future source plug-ins or format plug-ins.

See *Oracle8i interMedia Audio, Image, and Video User's Guide and Reference* for more information.

5.1 Prerequisites

You will need to include the following import statements in your Java file in order to run *inter*Media methods:

```
import java.sql.*;
import java.io.*;
import oracle.jdbc.driver.*;
import oracle.sql.*;
```

```
import oracle.ord.im.*;
```

The examples in this reference chapter are based on the assumption that the following operations have already been performed:

- A connection has been made to a table that contains a column of type ORDVideo.
- A local ORDVideo object named vidObj has been created and populated with data.

For examples of making a connection and populating a local object, see Section 2.3.2.

5.2 Reference Information

This section presents reference information on the methods that operate on ORDVideo objects.

checkProperties( )

Format

public boolean checkProperties(byte[ ] ctx[ ])

Description

Checks if the properties stored in the object attributes of the application ORDVideo object are consistent with the values stored in the application BLOB data.

Parameters

ctx[ ]
The format plug-in context information. It is set to NULL if there is no context information.

Return Value

This method returns TRUE if the attribute values stored in the object attributes are the same as the properties stored in the BLOB data; FALSE otherwise.

Exceptions

java.sql.SQLException

Example

```
byte[ ] ctx[ ] = new byte[4000][1];
if(vidObj.checkProperties(ctx))
     System.out.println("checkProperties successful");
```

where:

- ctx: contains the format plug-in context information.

clearLocal( )

Format

public void clearLocal( )

Description

Clears the source local field of the application ORDVideo object.

Parameters

None.

Return Value

None.

Exceptions

java.sql.SQLException

Example

```
vidObj.clearLocal( );
```

closeSource( )

Format

public int closeSource(byte[ ] ctx[ ])

Description

Closes the file source of the application ORDVideo object.

Parameters

ctx[ ]
The source plug-in context information. It is set to NULL if there is no context information.

Return Value

This method returns 0 if the operation is successful, or an integer greater than 0 in case of failure.

Exceptions

java.sql.SQLException

Example

```
byte[ ] ctx[ ] = new byte[4000][1];
int i = vidObj.closeSource(ctx);
if(i == 0)
     System.out.println("closeSource successful");
```

where:

- ctx: contains the source plug-in context information.

deleteContent( )

Format

public void deleteContent( )

Description

Deletes the media data in the application ORDVideo object.

Parameters

None.

Return Value

None.

Exceptions

java.sql.SQLException

Example

```
vidObj.deleteContent( );
```

export( )

Format

public void export (byte[ ] ctx[ ], String sourceType, String sourceLocation, String sourceName)

Description

Exports the data from the application ORDVideo object to the location specified in the parameters. The location is of the form:

sourceType: / / sourceLocation / sourceName

This method will work only if you are running Oracle8*i* database server release 8.1.7 or later.

See Chapter 4 of *Oracle8i interMedia Audio, Image, and Video User's Guide and Reference* for more information.

Parameters

ctx[ ]
The source plug-in context information. It is set to NULL if there is no context information.

sourceType
The source type to which the content will be exported.

sourceLocation
The source location to which the content will be exported.

sourceName
The source name to which the content will be exported.

Return Value

None.

Exceptions

java.sql.SQLException

Example

```
byte[ ] ctx[ ] = new byte[4000][1];
vidObj.export(ctx,"FILE","VIDEODIR","complete.dat");
```

where:

- ctx: contains the source plug-in context information.
- FILE: is the source type to which the content will be exported.
- VIDEODIR: is the location to which the content will be exported.
- complete.dat: is the file to which the content will be exported.

getAllAttributes( )

Format

public CLOB getAllAttributes(byte[ ] ctx[ ])

Description

Gets all the attributes from the application ORDVideo object and puts them in a CLOB.

Parameters

ctx[ ]
The format plug-in context information. It is set to NULL if there is no context information.

Return Value

This method returns a CLOB that contains the attribute values of the ORDVideo object.

Exceptions

java.sql.SQLException

Example

```
byte[ ] ctx[ ] = new byte[4000][1];
CLOB attributes = vidObj.getAllAttributes(ctx);
```

where:

- ctx: contains the format plug-in context information.

getAttribute( )

Format

public String getAttribute(byte[ ] ctx[ ], String name)

Description

Gets the value of a specified attribute from the application ORDVideo object as a String.

Parameters

ctx[ ]
The format plug-in context information. It is set to NULL if there is no context information.

name
The name of the attribute to get.

Return Value

This method returns the value of the specified attribute, as a String.

Exceptions

java.sql.SQLException

Example

```
byte[ ] ctx[ ] = new byte[4000][1];
String attribute = vidObj.getAttribute(ctx, video_duration);
```

where:

- ctx: contains the format plug-in context information.
- video_duration: is the attribute to get.

getBFILE( )

Format

public oracle.sql.BFILE getBFILE( )

Description

Gets the BFILE attribute of the application ORDVideo object.

Parameters

None.

Return Value

This method returns the BFILE.

Exceptions

java.sql.SQLException

Example

```
BFILE videoBFILE = vidObj.getBFILE( );
```

getBitRate( )

Format

public int getBitRate( )

Description

Gets the bit rate of the application ORDVideo object, in bits per second.

Parameters

None.

Return Value

This method returns the bit rate of the ORDVideo object.

Exceptions

java.sql.SQLException

Example

```
int bitRate = vidObj.getBitRate( );
```

getComments( )

Format

public oracle.sql.CLOB getComments( )

Description

Gets the comments from the application ORDVideo object and loads them into a CLOB.

Parameters

None.

Return Value

This method returns a CLOB that contains the comments from the ORDVideo object.

Exceptions

java.sql.SQLException

Example

```
CLOB comments = vidObj.getComments( );
```

getCompressionType( )

Format

public String getCompressionType( )

Description

Gets the compression type of the application ORDVideo object as a String.

Parameters

None.

Return Value

This method returns the compression type of the ORDVideo object, as a String.

Exceptions

java.sql.SQLException

Example

```
String compressionType = vidObj.getCompressionType( );
```

getContent( )

Format

public oracle.sql.BLOB getContent( )

Description

Gets the LOB locator from the application ORDVideo object.

Parameters

None.

Return Value

This method returns the LOB locator of the application object.

Exceptions

java.sql.SQLException

Example

```
BLOB localContent = vidObj.getContent( );
```

getContentInLob( )

Format

public BLOB getContentInLob(byte[ ] ctx[ ], String mimetype[ ], String format[ ])

Description

Gets the content of the application ORDVideo object and puts it in a BLOB.

Parameters

ctx[ ]
The source plug-in context information. It is set to NULL if there is no context information.

mimetype[ ]
The MIME type of the content returned, stored in mimetype[0]

format[ ]
The format of the content returned, stored in format[0].

Return Value

This method returns the LOB content in another LOB locator.

Exceptions

java.sql.SQLException

Example

```
byte[ ] ctx[ ] = new byte[4000][1];
String mimeType[ ] = new String[1];
mimeType[0] = "audio/x-aiff"
String format[ ] = new String[1];
format[0] = "AIFF"
BLOB localContent = vidObj.getContentInLob(ctx,mimeType,format);
```

where:

- ctx: contains the source plug-in context information.

- mimeType: is an array of Strings whose first value contains the MIME type.
- format: is an array of Strings whose first value contains the format.

getContentLength( )

Format

public int getContentLength( )

Description

Gets the content length of the media data in the application ORDVideo object.

Parameters

None.

Return Value

This method returns the content length of the media data.

Exceptions

java.sql.SQLException

Example

```
int contentLength = vidObj.getContentLength( );
```

getContentLength(byte[ ])

Format

public int getContentLength(byte[ ] ctx[ ])

Description

Gets the content length of the media data in the application ORDVideo object.

Parameters

ctx[ ]
The source plug-in context information. It is set to NULL if there is no context information.

Return Value

This method returns the content length of the media data.

Exceptions

java.sql.SQLException

Example

```
byte[ ] ctx[ ] = new byte[4000][1];
int contentLength = vidObj.getContentLength(ctx);
```

where:

- ctx: contains the source plug-in context information.

getDataInByteArray( )

Format

public byte[ ] getDataInByteArray( )

Description

Gets data from the LOB locator of the application ORDVideo object and puts it in a local byte array.

Parameters

None.

Return Value

This method returns the byte array from which the data will be read.

Exceptions

java.sql.SQLException

java.io.IOException

java.lang.OutOfMemoryError

Example

```
byte[ ] byteArr = vidObj.getDataInByteArray( );
```

getDataInFile( )

Format

public boolean getDataInFile(String filename)

Description

Gets data from the LOB locator of the application ORDVideo object and puts it in a local file.

Parameters

filename
The name of the file into which the data will be loaded.

Return Value

This method returns TRUE if loading is successful; FALSE otherwise.

Exceptions

java.sql.SQLException

java.io.IOException

Example

```
boolean load = vidObj.getDataInFile("output1.dat");
if(load)
     System.out.println("getDataInFile completed successfully");
else
     System.out.println("Error in getDataInFile");
```

where:

- output1.dat: is the file into which the data will be loaded.

getDataInStream( )

Format

public InputStream getDataInStream( )

Description

Gets data from the LOB locator of the application ORDVideo object and puts it in a local input stream.

Parameters

None.

Return Value

This method returns the input stream from which the data will be read.

Exceptions

java.sql.SQLException

Example

```
InputStream inpStream = vidObj.getDataInStream( );
```

getDescription( )

Format

public String getDescription( )

Description

Gets the description attribute of the application ORDVideo object.

Parameters

None.

Return Value

This method returns the description attribute as a String.

Exceptions

java.sql.SQLException

Example

```
String description = vidObj.getDescription( );
```

getFormat( )

Format

public String getFormat( )

Description

Gets the format attribute of the application ORDVideo object as a String.

Parameters

None.

Return Value

This method returns the format attribute as a String.

Exceptions

java.sql.SQLException

Example

```
String format = vidObj.getFormat( );
```

getFrameRate( )

Format

public int getFrameRate( )

Description

Get the frame rate in the application ORDVideo object, in frames per second.

Parameters

None.

Return Value

This method returns the frame rate of the ORDVideo object, as an integer.

Exceptions

java.sql.SQLException

Example

```
int frameRate = vidObj.getFrameRate( );
```

getFrameResolution( )

Format

public int getFrameResolution( )

Description

Gets the frame resolution of the application ORDVideo object, in pixels per inch.

Parameters

None.

Return Value

This method returns the frame resolution of the ORDVideo object.

Exceptions

java.sql.SQLException

Example

```
int frameResolution = vidObj.getFrameResolution( );
```

getHeight( )

Format

public int getHeight( )

Description

Gets the height of the application ORDVideo object.

Parameters

None.

Return Value

This method returns the height of the ORDVideo object.

Exceptions

java.sql.SQLException

Example

```
int height = vidObj.getHeight( );
```

getMimeType( )

Format

public String getMimeType( )

Description

Gets the MIME type of the application ORDVideo object as a String.

Parameters

None.

Return Value

This method returns the MIME type of the ORDVideo object, as a String.

Exceptions

java.sql.SQLException

Example

```
String mimeType = vidObj.gtMimeType( );
```

getNumberOfColors( )

Format

public int getNumberOfColors( )

Description

Gets the number of colors in the application ORDVideo object.

Parameters

None.

Return Value

This method returns the number of colors in the ORDVideo object, as an integer.

Exceptions

java.sql.SQLException

Example

```
int numberOfColors = vidObj.getNumberOfColors( );
```

getNumberOfFrames( )

Format

public int getNumberOfFrames( )

Description

Gets the number of frames in the application ORDVideo object.

Parameters

None.

Return Value

This method returns the number of frames in the ORDVideo object, as an integer.

Exceptions

java.sql.SQLException

Example

```
int numberOfFrames = vidObj.getNumberOfFrames( );
```

getSource( )

Format

public String getSource( )

Description

Gets the application ORDVideo object source information, including the source location, name, and type.

Parameters

None.

Return Value

This method returns a String containing the object source information.

Exceptions

java.sql.SQLException

Example

```
String source = viObj.getSource( );
```

getSourceLocation( )

Format

public String getSourceLocation( )

Description

Gets the application ORDVideo object source location as a String.

Parameters

None.

Return Value

This method returns a String containing the object source location.

Exceptions

java.sql.SQLException

Example

```
String location = vidObj.getSourceLocation( );
```

getSourceName( )

Format

public String getSourceName( )

Description

Gets the application ORDVideo object source name as a String.

Parameters

None.

Return Value

This method returns a String containing the object source name.

Exceptions

java.sql.SQLException

Example

```
String name = vidObj.getSourceName( );
```

getSourceType( )

Format

public String getSourceType( )

Description

Gets the application ORDVideo object source location as a String.

Parameters

None.

Return Value

This method returns a String containing the object source type.

Exceptions

java.sql.SQLException

Example

```
String type = vidObj.getSourceType( );
```

getUpdateTime( )

Format

public java.sql.Timestamp getUpdateTime( )

Description

Gets a Timestamp object that contains information on when the application ORDVideo object was most recently updated.

Parameters

None.

Return Value

This method returns a Timestamp object that contains the time of the most recent update.

Exceptions

java.sql.SQLException

Example

```
Timestamp time = audObj.getUpdateTime( );
```

getVideoDuration( )

Format

public int getVideoDuration( )

Description

Gets the video duration of the application ORDVideo object.

Parameters

None.

Return Value

This method returns the video duration of the ORDVideo object.

Exceptions

java.sql.SQLException

Example

```
int videoDuration = vidObj.getVideoDuration( );
```

getWidth( )

Format

public int getWidth( )

Description

Gets the width of the application ORDVideo object.

Parameters

None.

Return Value

This method returns the width of the ORDVideo object.

Exceptions

java.sql.SQLException

Example

```
int width = vidObj.getWidth( );
```

importData( )

Format

public void importData(byte[ ] ctx[ ])

Description

Imports data from an external source into the application ORDVideo object.

The srcType, srcLocation, and srcName attributes must all be defined for this method to work.

Parameters

ctx[ ]
The source plug-in context information. It is set to NULL if there is no context information.

Return Value

None.

Exceptions

java.sql.SQLException

Example

```
byte[ ] ctx[ ] = new byte[4000][1];
vidObj.importData(ctx);
```

where:

- ctx: contains the source plug-in information.

importFrom( )

Format

public void importFrom(byte[ ] ctx[ ], String sourceType, String sourceLocation, String sourceName)

Description

Imports data from an external source into the application ORDVideo object. The location of the external source is of the form:

sourceType: / / sourceLocation / sourceName

Parameters

ctx[ ]
The source plug-in context information. See Chapter 4 of *Oracle8i interMedia Audio, Image, and Video User's Guide and Reference* for more information.

sourceType
The source type from which the data will be imported.

sourceLocation
The source location from which the data will be imported.

sourceName
The source name from which the data will be imported.

Return Value

None.

Exceptions

java.sql.SQLException

Example

```
byte[ ] ctx[ ] = new byte[4000][1];
vidObj.importFrom(ctx,"FILE","VIDEODIR","testvid.dat");
```

where:

- ctx: contains the source plug-in context information.
- FILE: is the type of the source from which the data will be imported.
- VIDEODIR: is the location of the file from which the data will be imported.
- testvid.dat: is the file from which the data will be imported.

isLocal( )

Format

public boolean isLocal( )

Description

Checks if the application ORDVideo object local attribute is set.

Parameters

None.

Return Value

This method returns TRUE if the ORDVideo object local attribute is set; FALSE otherwise.

Exceptions

java.sql.SQLException

Example

```
if(vidObj.isLocal( ))
     System.out.println("local attribute is set to true");
else
     System.out.println("lcoal attribute is set to false");
```

loadDataFromByteArray( )

Format

public boolean loadDataFromByteArray(byte[ ] byteArr)

Description

Loads data from the local byte buffer into the database ORDVideo object LOB locator and into the application object. It also calls setLocal( ), which sets the local field of the application ORDVideo object, but not the database object, and setUpdateTime(null), which sets the updateTime attribute to the SYSDATE of the database server.

Parameters

byteArr
The name of the local byte array from which to load data.

Return Value

This method returns TRUE if loading is successful; FALSE otherwise.

Exceptions

java.sql.SQLException

java.io.IOException

Example

```
byte[ ] data = new byte[32000];
FileInputStream fStream = new FileInputStream("testvid.dat");
fStream.read(data,0,32300);
boolean success = vidObj.loadDataFromByteArray(data);
if(success)
     System.out.println("loadDataFromByteArray was successful");
else
     System.out.println("loadDataFromByteArray was unsuccessful");
```

where:

- data: is the local byte array from which the data will be loaded.

- testvid.dat: is a local file that contains 32,300 bytes of data.

loadDataFromFile( )

Format

public boolean loadDataFromFile(String filename)

Description

Loads data from the local file buffer into the database ORDVideo object LOB locator and into the application object. It also calls setLocal( ), which sets the local field of the application ORDVideo object, but not the database object, and setUpdateTime(null), which sets the updateTime attribute to the SYSDATE of the database server.

Parameters

filename
The name of the local file from which to load data.

Return Value

This method returns TRUE if loading is successful; FALSE otherwise.

Exceptions

java.sql.SQLException

java.io.IOException

Example

```
vidObj.loadDataFromFile("testvid.dat");
```

where:

- testvid.dat: is a local file that contains video data.

loadDataFromInputStream( )

Format

public boolean loadDataFromInputStream(InputStream inpStream)

Description

Loads data from the local input stream into the database ORDVideo object LOB locator and into the application object. It also calls setLocal( ), which sets the local field of the application ORDVideo object, but not the database object, and setUpdateTime(null), which sets the updateTime attribute to the SYSDATE of the database server.

Parameters

inpStream
The name of the local input stream from which to load data.

Return Value

This method returns TRUE if loading is successful; FALSE otherwise.

Exceptions

java.sql.SQLException

java.io.IOException

Example

```
FileInputStream fStream = new FileInputStream("testvid.dat");
vidObj.loadDataFromInputStream(fStream);
```

where:

- testvid.dat: is a local file that contains video data.
- fStream: is the local input stream that will load video data into the ORDVideo object.

openSource( )

Format

public int openSource(byte[ ] userarg, byte[ ] ctx[ ])

Description

Opens the file source of the application ORDVideo object.

Parameters

userarg
Permission-related parameters that are supplied by the user, such as READONLY. These may be used by user-defined source plug-ins.

ctx[ ]
The parameter for passing the context information of the caller. See Chapter 4 of *Oracle8i interMedia Audio, Image, and Video User's Guide and Reference* for more information.

Return Value

This method returns 0 if the operation is successful, or an integer greater than 0 in case of failure.

Exceptions

java.lang.Exception

Example

```
byte[ ] userarg = new byte[4000];
byte[ ] ctx[ ] = new byte[4000][1];
int i = audObj.openSource(userarg,ctx);
if(i == 0)
     System.out.println("openSource successful");
else
     System.out.println("openSource unsuccessful");
```

where:

- userarg: contains permission-related parameters.

- ctx: contains the source plug-in context information.

OrdVideo( )

Format

public OrdVideo( )

Description

Creates an ORDVideo object.

This method is the default ORDVideo constructor.

Parameters

None.

Return Value

None.

Exceptions

None.

Example

None.

processSourceCommand( )

Format

public byte[ ] processSourceCommand(byte[ ] ctx[ ], String command, String args, byte[ ] result[ ])

Description

Processes the specified command on the application ORDVideo object by calling the database processSourceCommand( ) method.

Parameters

ctx[ ]
The source plug-in context information. See Chapter 4 of *Oracle8i interMedia Audio, Image, and Video User's Guide and Reference* for more information.

command
The command to be executed.

args
The arguments of the command to be executed.

result[ ]
The results of the command.

Return Value

This method returns the results of executing the command.

Exceptions

java.sql.SQLException

Example

```
byte[ ] ctx[ ] = new byte[4000][1];
String command;
String arguments;
byte[ ] result;
//assign a command value to command
//assign any arguments to args
```

```
byte[ ] commandResults = vidObj.processSourceCommand(ctx,command,
     arguments,result);
```

where:

- ctx: contains the source plug-in information.
- command: is the command to be run.
- arguments: contains any arguments required by the command.
- result: is the results of the command.

processVideoCommand( )

Format

public byte[ ] processVideoCommand(byte[ ] ctx[ ], String command, String args, byte[ ] result[ ])

Description

Processes the specified command on the application ORDVideo object by calling the database processVideoCommand( ) method.

Parameters

ctx[ ]
The format plug-in context information. It is set to NULL if there is no context information.

command
The command to be executed.

args
The arguments of the command to be executed.

result[ ]
The results of the command.

Return Value

This method returns the results of executing the command.

Exceptions

java.sql.SQLException

Example

```
byte[ ] ctx[ ] = new byte[4000][1]
String command;
String arguments;
byte[ ] result;
//assign a command value to command
//assign any arguments to args
```

```
byte[ ] commandResults = vidObj.processVideoCommand(ctx,command,
     arguments,result);
```

where:

- ctx: contains the format plug-in information.
- command: is the command to be run.
- arguments: contains any arguments required by the command.
- result: is the results of the command.

readFromSource( )

Format

public int readFromSource(byte[ ] ctx[ ], int startpos, int numbytes, byte[ ] buffer)

Description

Reads data from the comments field of the application ORDVideo object.

Parameters

ctx[ ]
The source plug-in context information. See Chapter 4 of *Oracle8i interMedia Audio, Image, and Video User's Guide and Reference* for more information.

startpos
The initial position in the comments field.

numbytes
The number of bytes to be read.

buffer
The buffer into which the content will be read.

Return Value

This method returns the number of bytes read.

Exceptions

java.sql.SQLException

Example

```
byte[ ] ctx[ ] = new byte[4000][1];
byte[ ] commentBuffer = new byte[12];
int i = vidObj.readFromSource(ctx,0,12,commentBuffer);
```

where:

- ctx: contains the source plug-in context information.

- 0: is the position to begin reading from the comments field.
- 12: is the number of bytes to be read.
- commentBuffer: is the location to which the data will be read.

setBitRate( )

Format

public void setBitRate(int bitRate)

Description

Sets the bit rate of the application ORDVideo object.

setProperties( ) will automatically set this attribute; use this method only if you are not using setProperties( ). Also, this method will set only the attribute value; it does not change the media file itself.

Parameters

bitRate
The bit rate to be set in the ORDVideo object, in bits per second.

Return Value

None.

Exceptions

java.sql.SQLException

Example

```
vidObj.setBitRate(1500);
```

where:

- 1500: is the bit rate, in bits per second.

setComments( )

Format

public void setComments(oracle.sql.CLOB comments)

Description

Sets the comments in the application ORDVideo object.

The comments attribute is reserved for use by *inter*Media. You can set your own value, but it could be overwritten by *inter*Media Annotator.

Parameters

comments
A CLOB that contains comments for the ORDVideo object.

Return Value

None.

Exceptions

java.sql.SQLException

Example

```
vidObj.setComments(commentsData);
```

where:

- commentsData: is a CLOB that contains comments to be set.

setCompressionType( )

Format

public void setCompressionType(String CompressionType)

Description

Sets the compression type of the application ORDVideo object.

setProperties( ) will automatically set this attribute; use this method only if you are not using setProperties( ). Also, this method will set only the attribute value; it does not change the media file itself.

Parameters

CompressionType
The compression type of the ORDVideo object, as a String.

Return Value

None.

Exceptions

java.sql.SQLException

Example

```
vidObj.setCompressionType("Cinepak");
```

where:

- Cinepak: is the compression type to be set.

setDescription( )

Format

public void setDescription(String description)

Description

Sets the description attribute of the application ORDVideo object.

Parameters

description
A String that describes the contents of the ORDVideo object.

Return Value

None.

Exceptions

java.sql.SQLException

Example

```
vidObj.setDescription("My video file");
```

where:

- My video file: is the description to be set in the object.

setFrameRate( )

Format

public void setFrameRate(int frameRate)

Description

Sets the frame rate of the application ORDVideo object.

setProperties( ) will automatically set this attribute; use this method only if you are not using setProperties( ). Also, this method will set only the attribute value; it does not change the media file itself.

Parameters

frameRate
The frame rate to be set in the ORDVideo object, in frames per second.

Return Value

None.

Exceptions

java.sql.SQLException

Example

```
vidObj.setFrameRate(5);
```

where:

- 5: is the frame rate, in frames per second.

setFrameResolution( )

Format

public void setFrameResolution(int frameResolution)

Description

Sets the frame resolution of the application ORDVideo object.

setProperties( ) will automatically set this attribute; use this method only if you are not using setProperties( ). Also, this method will set only the attribute value; it does not change the media file itself.

Parameters

frameResolution
The frame resolution to be set in the ORDVideo object.

Return Value

None.

Exceptions

java.sql.SQLException

Example

```
vidObj.setFrameResolution(4);
```

where:

- 4: is the frame resolution.

setFormat( )

Format

public void setFormat(String format)

Description

Sets the format attribute of the application ORDVideo object.

setProperties( ) will automatically set this attribute; use this method only if you are not using setProperties( ). Also, this method will set only the attribute value; it does not change the media file itself.

Parameters

format
The format of the contents of the ORDVideo object, as a String.

Return Value

None.

Exceptions

java.sql.SQLException

Example

```
vidObj.setFormat("MOOV");
```

where:

- MOOV: is the format to be set.

setHeight( )

Format

public void setHeight(int height)

Description

Sets the height of the application ORDVideo object.

setProperties( ) will automatically set this attribute; use this method only if you are not using setProperties( ). Also, this method will set only the attribute value; it does not change the media file itself.

Parameters

height
The height of the ORDVideo object, in pixels.

Return Value

None.

Exceptions

java.sql.SQLException

Example

```
vidObj.setHeight(24);
```

where:

- 24: is the height, in pixels.

setKnownAttributes( )

Format

public void setKnownAttributes(String knownformat, String knownwidth, String knownheight,
int knownframeresolution, int knownframerate, int knownvideoduration,
int knownnumberoffframes, String knowncompressiontype,
int knownnumberofcolors, int knownbitrate

Description

Sets the known attributes of the application ORDVideo object.

setProperties( ) will automatically set these attributes; use this method only if you are not using setProperties( ). Also, this method will set only the attribute values; it does not change the media file itself.

Parameters

knownformat
The video data format.

knownwidth
The video width.

knownheight
The video height.

knownframeresolution
The frame resolution.

knownframerate
The frame rate.

knownvideoduration
The video duration.

knownnumberoffframes
The number of frames.

knowncompressiontype
The compression type.

knownnumberofcolors
The number of colors.

knownbitrate
The bit rate.

Return Value

None.

Exceptions

java.sql.SQLException

Example

```
vidObj.setKnownAttributes("MOOV",1,2,4,5,20,8,"Cinepak",256,1500);
```

where:

- MOOV: is the format.
- 1: is the width, in pixels.
- 2: is the height, in pixels.
- 4: is the frame resolution.
- 5: is the frame rate, in frames per second.
- 20: is the video duration.
- 8: number of frames.
- Cinepak: is the compression type.
- 256: is the number of colors.
- 1500: is the bit rate, in bits per second.

setLocal( )

Format

public void setLocal( )

Description

Sets the source local field of the application ORDVideo object.

Parameters

None.

Return Value

None.

Exceptions

java.sql.SQLException

Example

```
vidObj.setLocal( );
```

setMimeType( )

Format

public void setMimeType(String MimeType)

Description

Sets the MIME type of the application ORDVideo object.

setProperties( ) will automatically set this attribute; use this method only if you are not using setProperties( ). Also, this method will set only the attribute value; it does not change the media file itself.

Parameters

MimeType
The MIME type of the contents of the ORDVideo object, as a String.

Return Value

None.

Exceptions

java.sql.SQLException

Example

```
vidObj.setMimeType("video/avi");
```

where:

- video / avi: is the MIME type to be set.

setNumberOfColors( )

Format

public void setNumberOfColors(int numberOfColors)

Description

Sets the number of colors in the application ORDVideo object.

setProperties( ) will automatically set this attribute; use this method only if you are not using setProperties( ). Also, this method will set only the attribute value; it does not change the media file itself.

Parameters

numberOfColors
The number of colors to be set in the ORDVideo object.

Return Value

None.

Exceptions

java.sql.SQLException

Example

```
vidObj.setNumberOfColors(256);
```

where:

- 256: is the number of colors to be set.

setNumberOfFrames( )

Format

public void setNumberOfFrames(int numberOfFrames)

Description

Sets the number of frames in the application ORDVideo object.

setProperties( ) will automatically set this attribute; use this method only if you are not using setProperties( ). Also, this method will set only the attribute value; it does not change the media file itself.

Parameters

numberOfFrames
The number of frames to be set in the ORDVideo object.

Return Value

None.

Exceptions

java.sql.SQLException

Example

```
vidObj.setNumberOfFrames(8);
```

where:

- 8: is the number of frames to be set.

setProperties(byte[ ][ ])

Format

public void setProperties(byte[ ] ctx[ ])

Description

Reads the video data, extract the attributes, and sets the properties in the application ORDVideo object.

The properties to be set 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. It is set to NULL if there is no context information.

Return Value

None.

Exceptions

java.sql.SQLException

Example

```
byte[ ] ctx[ ] = new byte[4000][1];
vidObj.setProperties(ctx);
```

where:

- ctx: contains the format plug-in context information.

setProperties(byte[ ][ ], boolean)

Format

public void setProperties(byte[ ] ctx[ ], boolean setComments)

Description

Reads the video data, extract the attributes, and sets the properties in the application ORDVideo object.

The properties to be set 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. It is set to NULL if there is no context information.

setComments
A Boolean value to determine whether or not to set the comments in the ORDAudio object. If TRUE, the comments field is populated with a set of format and application properties of the audio object in XML. If FALSE, the comments field remains unpopulated. The default value is FALSE.

Return Value

None.

Exceptions

java.sql.SQLException

Example

```
byte[ ] ctx[ ] = new byte[4000][1];
vidObj.setProperties(ctx,true);
```

where:

- ctx: contains the format plug-in context information.

- true: indicates that the comments field will be populated.

setSource( )

Format

public void setSource(String sourceType, String sourceLocation, String sourceName)

Description

Sets the application ORDVideo object source information.

Parameters

sourceType
The type of the source.

sourceLocation
The location of the source.

sourceName
The name of the source.

Return Value

None.

Exceptions

java.sql.SQLException

Example

```
vidObj.setSource("LOCAL","VIDEODIR","video.dat");
```

where:

- LOCAL: is the source type.
- VIDEODIR: is the source location.
- video.dat: is the source name.

setUpdateTime( )

Format

public void setUpdateTime(java.sql.Timestamp currentTime)

Description

Sets the update time in the application ORDVideo object to the current time.

setProperties( ) will automatically set this attribute; use this method only if you are not using setProperties( ). Also, this method will set only the attribute value; it does not change the media file itself.

Parameters

currentTime
The current time, which will be set in the ORDVideo object. This value should be NULL; the method will then use the SYSDATE of the database server.

Return Value

None.

Exceptions

java.sql.SQLException

Example

```
vidObj.setUpdateTime(null);
```

setVideoDuration( )

Format

public void setVideoDuration(int videoDuration)

Description

Sets the video duration of the application ORDVideo object.

setProperties( ) will automatically set this attribute; use this method only if you are not using setProperties( ). Also, this method will set only the attribute value; it does not change the media file itself.

Parameters

videoDuration
The video duration of the ORDVideo object.

Return Value

None.

Exceptions

java.sql.SQLException

Example

```
vidObj.setVideoDuration(20);
```

where:

- 20: is the video duration to be set.

setWidth( )

Format

public void setWidth(int width)

Description

Sets the width of the application ORDVideo object.

setProperties( ) will automatically set this attribute; use this method only if you are not using setProperties( ). Also, this method will set only the attribute value; it does not change the media file itself.

Parameters

width
The width value to be set, in pixels.

Return Value

None.

Exceptions

java.sql.SQLException

Example

```
vidObj.setWidth(24);
```

where:

- 24: is the width, in pixels.

trimSource( )

Format

public int trimSource(byte[ ] ctx[ ], int newLen)

Description

Trims the application ORDVideo file source to the given length.

Parameters

ctx[ ]
The source plug-in context information. It is set to NULL if there is no context information.

newLen
The length to which the source will be trimmed.

Return Value

This method returns 0 if the operation is successful, or an integer greater than 0 in case of failure.

Exceptions

java.sql.SQLException

Example

```
byte[ ] ctx[ ] = new byte[4000][1];
int i = vidObj.trimSource(ctx,10);
if (i == 0)
     System.out.println("trimSource successful");
else
     System.out.println("trimSource unsuccessful");
```

where:

- ctx: contains the source plug-in context information.
- 10: is the new length of the source.

writeToSource( )

Format

public int writeToSource(byte[ ] ctx[ ], int startpos, int numbytes, byte[ ] buffer)

Description

Writes data to the comments field of the application ORDVideo object.

Parameters

ctx[ ]
The source plug-in context information. See Chapter 4 of *Oracle8i interMedia Audio, Image, and Video User's Guide and Reference* for more information.

startpos
The initial position in the comments field.

numbytes
The number of bytes to be written.

buffer
The buffer containing the content to be written.

Return Value

This method returns the number of bytes written.

Exceptions

java.sql.SQLException

Example

```
byte[ ] ctx[ ] = new byte[4000][1];
byte[ ] data = new byte[20];
//populate data with 20 bytes of content
int i = vidObj.writeToSource(ctx,1,20,data);
```

where:

- ctx: contains the source plug-in context information.

- 1: is the position in the comments field where writing will begin.
- 20: is the number of bytes to be written.
- data: contains the content to be written.

A

Running Java Classes Examples

Three sample files (programs written in Java) are provided in the installation of *inter*Media Java Classes. These files provide examples of how to build Java applications with *inter*Media. They demonstrate loading data from various sources into database objects, downloading data from database objects to the file system, and extracting and displaying metadata from the media content.

The names of the Java sample files are as follows:

- For audio: AudioExample.java
- For image: ImageExample.java
- For video: VideoExample.java

In order to run the Java sample files included with *inter*Media Java Classes, you must perform the following operations:

1. Install Oracle8*i* with Oracle *inter*Media.

   You must have the Oracle8*i* database server that includes Oracle *inter*Media installed on a server machine.

2. Check the values of local variables.

   All users must make sure that classes111.zip, ordim817.zip, the JDK classes, and the SQLJ runtime.zip files are included in the CLASSPATH variable on the local machine, and the local directory containing the javac and java commands is included in the PATH variable on the local machine.

   Solaris users must make sure the directory that contains ocijdbc8.so is included in the LD_LIBRARY_PATH variable.

   Windows NT users must make sure that the directory that contains ocijdbc8.dll is included in the PATH variable.

3. Compile the Java file on your local machine.

   Using version 1.1.6 of the JDK, compile the sample programs using the appropriate command:

   - For audio: `javac AudioExample.java`
   - For image: `javac ImageExample.java`
   - For video: `javac VideoExample.java`

4. Connect to your database and run the SQL script that corresponds to your Java file.

   In order for the sample programs to run, your database must include tables that contain a column of the appropriate *inter*Media object type. The *inter*Media Java Classes installation includes three SQL files that contain commands to create a new user and create a table, and add some sample data to the table.

   The names of the SQL scripts are as follows:

   - For audio: AudioExample.sql
   - For image: ImageExample.sql
   - For video: VideoExample.sql

   Note: The SQL script connects to the database as the user *system*, with a password of *manager*. Edit the SQL files to change the password or remove the connect statement before running the script.

   Also, edit the directory path to reflect your schema.

5. Run the compiled Java program.

   Run the sample program using the appropriate command:

   - For audio: `java AudioExample`
   - For image: `java ImageExample`
   - For video: `java VideoExample`

For more information on the sample files, see the appropriate readme file.

B

Exceptions and Errors

This appendix contains information on the exceptions and errors that can be raised by i*nter*Media Java Classes.

B.1 Exception Class

The Exception class (and its subclasses, including SQLException) indicates conditions of interest to the user.

```
public class java.lang.Exception extends java.lang.Throwable {
     //Constructs an Exception with no detailed message
     public Exception();

     //Constructs an Exception with a detailed message
     public Exception(String s);
}
```

B.2 IOException Class

The IOException class signals that an I/O exception of some sort has occurred.

```
public class java.io.IOException extends java.lang.Exception {
     //Constructs a FileNotFoundException with no detailed message
     public IOException();

     //Constructs a FileNotFoundException with the specified detailed message
     public IOException(String s);
}
```

B.3 OutOfMemoryError Class

The OutOfMemoryError class signals that the Java Virtual Machine cannot allocate an object because it is both out of memory and unable to make more memory available through garbage collecting (that is, through deleting objects that are no longer being used).

```
public class java.lang.OutOfMemoryError extends java.lang.VirtualMachineError {
     //Constructs an OutOfMemoryError with no detailed message
     public OutOfMemoryError();

     //Constructs an OutOfMemoryError with a detailed message
     public OutOfMemoryError(string s);
}
```

B.4 SQLException Class

The SQLException class provides information on a database access error.

```
public class java.sql.SQLException extends java.lang.Exception {
     //The following four methods are public constructors:

     //Constructs a fully specified SQLException
     public SQLException(String reason, String SQLState, int vendorCode);

     //Constructs a SQLException with venderCode value of 0
     public SQLException(String reason, String SQLState);

     //Constructs a SQLException with vendorCode value of 0 and a null SQLState
     public SQLException(String reason);

     //Constructs a SQLException with vendorCode of 0, a null SQLState. and a
     //null reason
     public SQLException();

     //The following four methods are public instance methods:

     //Gets the vendor-specific exception code
     public int getErrorCode();

     //Gets the exception connected to this one
     public SQLException getNextException();

     //Gets the SQL state
     public String getSQLState();
```

```
    //Adds a SQLException to the end
    public synchronized void setNextException(SQLException ex);
}
```

C

Deprecated Methods

The following list shows a list of the methods that have been deprecated since release 8.1.5 of Oracle8*i* *inter*Media Audio, Image, and Video Java Client.

- public void appendToComments(int amount, String buffer)
- public int compareComments(CLOB dest, int amount, int start_in_comment, int start_in_compare_comment)
- public CLOB copyCommentsOut(CLOB dest, int amount, int from_loc, int to_loc)
- public void deleteComments( )
- public int eraseFromComments(int amount, int offset)
- public void flush( ) throws SQLException
- public String getAllAttributesAsString(byte[ ] ctx)
- public int getAudioDuration(byte[ ] ctx)
- public int getCommentLength( )
- public String getCommentsAsString( )
- public String getCompressionType(byte[ ] ctx)
- public byte[ ] getData(String tableName, String columnName, String condition)
- public String getEncoding(byte[ ] ctx)
- public String getFormat(byte[ ] ctx)
- public int getNumberOfChannels(byte[ ] ctx)
- public int getSampleSize(byte[ ] ctx)
- public int getSamplingRate(byte[ ] ctx)

- public boolean loadComments(String filename)
- public void loadCommentsFromFile(String loc, String fileName, int amount, int from_loc, int to_loc)
- public boolean loadData(String fileName, String tableName, String columnName, String condition)
- public int locateInComment(String pattern, int offset, int occurrence)
- public OrdAudio(Connection the_connection)
- public OrdVideo(Connection the_connection)
- public String readFromComments(int offset, int amount)
- public void refresh(boolean forUpdate)
- public void setBindParams(String tableName, String columnName, String condition)
- public void trimComments(int newlen)
- public void writeToComments(int offset, int amount, String buffer)

Index

I

L

M

O

P

R

S

T

W