

Oracle® Multimedia

User's Guide



18c
E84257-01
February 2018

The Oracle logo, consisting of a solid red square with the word "ORACLE" in white, uppercase, sans-serif font centered within it.

ORACLE®

Oracle Multimedia User's Guide, 18c

E84257-01

Copyright © 1999, 2018, Oracle and/or its affiliates. All rights reserved.

Primary Author: Sue Pelski

Contributors: Melliyal Annamalai, Susan Mavris, Valarie Moore, James Steiner

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

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

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

U.S. GOVERNMENT END USERS: Oracle programs, including any operating system, integrated software, any programs installed on the hardware, and/or documentation, delivered to U.S. Government end users are "commercial computer software" pursuant to the applicable Federal Acquisition Regulation and agency-specific supplemental regulations. As such, use, duplication, disclosure, modification, and adaptation of the programs, including any operating system, integrated software, any programs installed on the hardware, and/or documentation, shall be subject to license terms and license restrictions applicable to the programs. No other rights are granted to the U.S. Government.

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

Oracle and Java are registered trademarks of Oracle and/or its affiliates. Other names may be trademarks of their respective owners.

Intel and Intel Xeon are trademarks or registered trademarks of Intel Corporation. All SPARC trademarks are used under license and are trademarks or registered trademarks of SPARC International, Inc. AMD, Opteron, the AMD logo, and the AMD Opteron logo are trademarks or registered trademarks of Advanced Micro Devices. UNIX is a registered trademark of The Open Group.

This software or hardware and documentation may provide access to or information about content, products, and services from third parties. Oracle Corporation and its affiliates are not responsible for and expressly disclaim all warranties of any kind with respect to third-party content, products, and services unless otherwise set forth in an applicable agreement between you and Oracle. Oracle Corporation and its affiliates will not be responsible for any loss, costs, or damages incurred due to your access to or use of third-party content, products, or services, except as set forth in an applicable agreement between you and Oracle.

Contents

Preface

Audience	xi
Documentation Accessibility	xi
Related Documents	xii
Conventions	xii

Changes in This Release for Oracle Multimedia User's Guide

Changes in Oracle Database Release 18c, Version 18.1	xiv
--	-----

1 Introduction to Oracle Multimedia

1.1 Oracle Multimedia Architecture	1-2
1.2 Object Relational Technology	1-3
1.3 Oracle Multimedia Capabilities	1-4
1.3.1 Oracle Multimedia Support for CDBs	1-7
1.3.2 Data Guard Rolling Upgrade Support for Oracle Multimedia	1-7
1.4 Audio Concepts	1-8
1.4.1 Digitized Audio	1-8
1.4.2 Audio Components	1-8
1.5 ORDDoc or Heterogeneous Media Data Concepts	1-9
1.5.1 Digitized Heterogeneous Media Data	1-9
1.5.2 Heterogeneous Media Data Components	1-9
1.6 Image Concepts	1-10
1.6.1 Digitized Images	1-10
1.6.2 Image Components	1-10
1.6.3 Metadata in Images	1-11
1.6.4 Medical Imaging	1-11
1.6.5 Metadata Extraction	1-12
1.6.6 Image Processing	1-12
1.7 Video Concepts	1-13
1.7.1 Digitized Video	1-13
1.7.2 Video Components	1-13

1.8	Loading Multimedia Data	1-14
1.9	Multimedia Storage and Querying	1-14
1.9.1	Storing Multimedia Data	1-15
1.9.2	Querying Multimedia Data	1-16
1.10	Accessing Multimedia Data	1-16

2 Oracle Multimedia Application Development

2.1	Developing Multimedia Applications Using SQL Developer	2-2
2.2	Developing Multimedia Applications Using Application Express	2-2
2.3	Developing Multimedia Applications Using Java and JDBC	2-3
2.3.1	Media Upload in Java	2-3
2.3.2	Retrieval of Image Properties in Java	2-5
2.3.3	Thumbnail Image Creation in Java	2-6
2.3.4	Handling Oracle Multimedia Exceptions in Java	2-7
2.3.4.1	Handling the Setting of Properties for Unknown Image Formats in Java	2-8
2.3.4.2	Handling Image Processing for Unknown Image Formats in Java	2-9
2.4	Developing Multimedia Applications Using PL/SQL	2-9
2.4.1	Setting Up Directory Objects for Media Import and Export	2-10
2.4.2	Media Upload in PL/SQL	2-10
2.4.3	Media Query in PL/SQL	2-11
2.4.4	Media Download in PL/SQL	2-12
2.4.5	Handling Oracle Multimedia Exceptions in PL/SQL	2-12
2.4.5.1	Handling the Setting of Properties for Unknown Image Formats in PL/SQL	2-13
2.4.5.2	Handling Image Processing for Unknown Image Formats in PL/SQL	2-14
2.5	Developing PL/SQL Web Applications	2-14
2.5.1	Using the PL/SQL Gateway and PL/SQL Web Toolkit	2-15

3 Oracle Multimedia PL/SQL Photo Album Sample Application

3.1	Overview of the PL/SQL Photo Album Sample Application	3-2
3.2	Running the PL/SQL Photo Album Sample Application	3-4
3.3	Description of the PL/SQL Photo Album Sample Application	3-5
3.3.1	Browsing the Photo Album	3-6
3.3.2	Adding Images to the Photo Album	3-10
3.3.3	Searching for Images by Keyword or Phrase	3-14
3.3.4	Viewing Full-Size Images	3-15
3.3.5	Examining Image Metadata	3-17
3.3.6	Writing New XMP Metadata to Images	3-18

4 Oracle Multimedia Code Wizard Sample Application for the PL/SQL Gateway

4.1	Running the Code Wizard Sample Application	4-1
4.2	Description of the Code Wizard Sample Application	4-2
4.2.1	Creating a New DAD or Choosing an Existing DAD	4-2
4.2.2	Authorizing a DAD	4-4
4.2.3	Creating and Testing Media Upload and Retrieval Procedures	4-6
4.2.4	Creating a Media Upload Procedure	4-8
4.2.5	Creating a Media Retrieval Procedure	4-13
4.2.6	Using the PL/SQL Gateway Document Table	4-17
4.2.7	How Time Zone Information Is Used to Support Browser Caching	4-18
4.3	Sample Session: Using Images	4-19
4.4	Known Restrictions of the Oracle Multimedia Code Wizard	4-28

5 Working with Metadata in Oracle Multimedia Images

5.1	Metadata Concepts	5-1
5.2	Oracle Multimedia Image Metadata Concepts	5-2
5.3	Image File Formats	5-2
5.4	Image Metadata Formats	5-2
5.4.1	EXIF	5-2
5.4.2	IPTC-IIM	5-3
5.4.3	XMP	5-3
5.5	Representing Metadata Outside Images	5-3
5.6	Oracle Multimedia Image Metadata Examples	5-4
5.6.1	Creating a Table for Metadata Storage	5-4
5.6.2	Extracting Image Metadata	5-5
5.6.3	Embedding Image Metadata	5-6
5.7	Metadata References	5-8

6 Oracle Multimedia Tuning Tips for DBAs

6.1	Understanding the Performance Profile of Oracle Multimedia Operations	6-1
6.2	Choosing LOB Storage Parameters for Multimedia LOBs	6-4
6.2.1	SecureFiles LOBs	6-5
6.2.2	TABLESPACE	6-5
6.2.3	CACHE, NOCACHE, and CACHE READS	6-5
6.2.4	LOGGING and NOLOGGING	6-5

6.2.5	Example of Setting LOB Storage Options	6-6
6.3	Setting Database Initialization Parameters	6-7

A Managing Oracle Multimedia Installations

A.1	Oracle Multimedia Installed Users and Privileges	A-1
A.2	Installing and Configuring Oracle Multimedia	A-2
A.2.1	Preinstallation Steps	A-3
A.2.2	Installation and Configuration Steps	A-4
A.3	Verifying an Installed Version of Oracle Multimedia	A-5
A.4	Upgrading an Installed Version of Oracle Multimedia	A-5
A.5	Downgrading an Installed Version of Oracle Multimedia	A-5

B Extending Oracle Multimedia

B.1	Supporting Other External Sources	B-1
B.1.1	External Source Packages	B-2
B.1.1.1	ORDPLUGINS.ORDX_FILE_SOURCE Package	B-2
B.1.1.2	ORDPLUGINS.ORDX_HTTP_SOURCE Package	B-4
B.1.1.3	Extending Oracle Multimedia to Support a New Data Source	B-6
B.2	Supporting Other Media Data Formats	B-8
B.2.1	Supporting Other ORDAudio Data Formats	B-9
B.2.1.1	ORDPLUGINS.ORDX_DEFAULT_AUDIO Package	B-9
B.2.1.2	Extending Oracle Multimedia to Support a New Audio Data Format	B-10
B.2.2	Supporting Other ORDDoc Data Formats	B-12
B.2.2.1	ORDPLUGINS.ORDX_DEFAULT_DOC Package	B-12
B.2.2.2	Extending Oracle Multimedia to Support a New ORDDoc Data Format	B-13
B.2.3	Supporting Other Video Data Formats	B-13
B.2.3.1	ORDPLUGINS.ORDX_DEFAULT_VIDEO Package	B-14
B.2.3.2	Extending Oracle Multimedia to Support a New Video Data Format	B-15
B.2.4	Supporting Other Image Data Formats	B-17
B.3	Supporting Media Data Processing	B-17
B.3.1	Supporting Audio Data Processing	B-17
B.3.2	Supporting Video Data Processing	B-17

C Oracle Multimedia Sample Applications

C.1	Oracle Multimedia ORDImage OCI C Sample Application	C-1
-----	---	-----

Glossary

Index

List of Examples

2-1	Image Query (Height, Width, and MimeType Attributes)	2-12
2-2	Audio Query (MimeType Attribute)	2-12
2-3	Video Query (MimeType Attribute)	2-12
2-4	URL Format to Invoke mod_plsql in a Web Browser	2-17
2-5	URL Format to Invoke mod_plsql for the Photo Album Application	2-17
3-1	Procedure view_album	3-8
3-2	Procedure print_album	3-8
3-3	Procedure print_image_link	3-9
3-4	Procedure deliver_media	3-9
3-5	Procedure print_upload_form	3-11
3-6	Procedure insert_new_photo	3-11
3-7	Procedure view_entry	3-16
3-8	Procedure view_metadata	3-18
3-9	Procedure print_metadata	3-18
3-10	Procedure write_metadata	3-20
3-11	Procedure search_metadata	3-22
B-1	Package Body for Extending Support to a New Data Source	B-6
B-2	Package Body for Extending Support to a New Audio Data Format	B-11
B-3	Package Body for Extending Support to a New ORDDoc Data Format	B-13
B-4	Package Body for Extending Support to a New Video Data Format	B-15

List of Figures

1-1	Oracle Multimedia Architecture	1-2
2-1	Components of a PL/SQL Web Application	2-16
3-1	View album Page with Uploaded Images	3-7
3-2	Completed Upload photo Page	3-10
3-3	Search album Page Showing Results	3-15
3-4	View entry Page with a Full-Size Image	3-16
3-5	View metadata Page with Metadata for an Uploaded Image	3-17
3-6	Completed Write XMP metadata Page with XMP Metadata for an Uploaded Image	3-19
3-7	Completed Search metadata Page for an Uploaded Image	3-22
4-1	Main Menu for the Code Wizard	4-4
4-2	Authorize the SCOTTCW DAD	4-5
4-3	List of Authorized DADs	4-6
4-4	Use the SCOTTCW DAD	4-7
4-5	Create a Media Upload Procedure	4-8
4-6	Media Upload Step 1: Select Database Table and Procedure Type	4-8
4-7	Media Upload Step 2: Select PL/SQL Gateway Document Upload Table	4-9
4-8	Media Upload Step 3: Select Data Access and Media Column(s)	4-10
4-9	Media Upload Step 4: Select Additional Columns and Procedure Name	4-11
4-10	Media Upload Step 5: Review Selected Options	4-12
4-11	Compiled Upload Procedure with Success Message	4-12
4-12	Template Upload Form for the Code Wizard	4-13
4-13	Create a Media Retrieval Procedure	4-14
4-14	Media Retrieval Step 1: Select Database Table and Procedure Type	4-14
4-15	Media Retrieval Step 2: Select Media Column and Key Column	4-15
4-16	Media Retrieval Step 3: Select Procedure Name and Parameter Name	4-16
4-17	Media Retrieval Step 4: Review Selected Options	4-16
4-18	Compiled Retrieval Procedure with Success Message	4-17

List of Tables

6-1	Performance Profile For All Multimedia Types	6-2
6-2	Performance Profile for ORD_IMAGE PL/SQL Package Functions and Procedures	6-2
6-3	Performance Profile For ORDImage Methods	6-3
6-4	Performance Profile for ORD_DICOM PL/SQL Package Functions and Procedures	6-3
6-5	Performance Profile For ORDDicom Methods	6-3
6-6	Performance Profile for ORD_AUDIO, ORD_DOC, and ORD_VIDEO PL/SQL Package Procedures	6-4
6-7	Performance Profile For ORDAudio, ORDDoc, and ORDVideo Methods	6-4
A-1	Installed Database Users	A-1
A-2	User Accounts and Default Passwords	A-2
B-1	Methods Supported in the ORDPLUGINS.ORDX_FILE_SOURCE Package	B-4
B-2	Methods Supported in the ORDPLUGINS.ORDX_HTTP_SOURCE Package	B-5
B-3	Methods Supported in the ORDPLUGINS.ORDX_DEFAULT_AUDIO Package	B-10
B-4	Method Supported in the ORDPLUGINS.ORDX_DEFAULT_DOC Package	B-12
B-5	Methods Supported in the ORDPLUGINS.ORDX_DEFAULT_VIDEO Package	B-14
C-1	Oracle Multimedia Sample Applications in Oracle Database Examples Media	C-1

Preface

This guide describes how to use Oracle Multimedia, which ships with Oracle Database. It describes the management and integration of audio, image, and video, or other heterogeneous media data with other Oracle tools and software, and with third-party tools and software.

In Oracle Database 11g Release 1 (11.1), the name Oracle *interMedia* was changed to Oracle Multimedia. The feature remains the same, only the name has changed.

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

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

Audience

This guide is for application developers and database administrators who are interested in storing, retrieving, and manipulating audio, image, video, and heterogeneous media data in a database, including developers of audio, heterogeneous media data, image, and video specialization options. After familiarizing yourself with the concepts presented in this guide, consult *Oracle Multimedia Reference* for API information.

Documentation Accessibility

For information about Oracle's commitment to accessibility, visit the Oracle Accessibility Program website at <http://www.oracle.com/pls/topic/lookup?ctx=acc&id=docacc>.

Access to Oracle Support

Oracle customers that have purchased support have access to electronic support through My Oracle Support. For information, visit <http://www.oracle.com/pls/topic/lookup?ctx=acc&id=info> or visit <http://www.oracle.com/pls/topic/lookup?ctx=acc&id=trs> if you are hearing impaired.

Related Documents



Note:

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

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

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

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

- *Oracle Multimedia Reference*
- *Oracle Multimedia DICOM Developer's Guide (Deprecated)*
- *Oracle Call Interface Programmer's Guide*
- *Oracle Database Development Guide*
- *Oracle Database SecureFiles and Large Objects Developer's Guide*
- *Oracle Database Concepts*
- *Oracle Database PL/SQL Language Reference*
- *Oracle Database Java Developer's Guide*
- *Oracle Database Error Messages*

For more information about using JDBC, see *Oracle Database JDBC Developer's Guide*.

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

For more information about Java, see the API documentation provided by Oracle.

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

Conventions

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

The following text conventions are also used in this guide:

Convention	Meaning
boldface	Boldface type indicates graphical user interface elements associated with an action, or terms defined in text or the glossary.

Convention	Meaning
<i>italic</i>	Italic type indicates book titles, emphasis, or placeholder variables for which you supply particular values.
monospace	Monospace type indicates commands within a paragraph, URLs, code in examples, text that appears on the screen, or text that you enter.

Changes in This Release for Oracle Multimedia User's Guide

This preface lists changes in *Oracle Multimedia User's Guide*.

Changes in Oracle Database Release 18c, Version 18.1

The following are changes in *Oracle Multimedia User's Guide* for Oracle Database 18c.

Desupported Features

The following features are desupported in this release:

- Oracle Multimedia Java client proxy classes, including DICOM proxy classes, and Oracle Multimedia servlet and JSP classes

To develop Java applications that manage multimedia content within Oracle Databases, Oracle recommends that you embed PL/SQL blocks in Java.

- Oracle Multimedia DICOM protocol, DICOM mid-tier support, and DICOM and UCM integration

There is no replacement for Oracle Multimedia DICOM. Oracle recommends that you replace Oracle Multimedia DICOM by using Oracle SecureFiles with third-party products for DICOM functionality. For example: Use third-party DICOM features to carry out metadata management, DICOM image conversion, and so on.

See Also:

- *Oracle Database PL/SQL Language Reference*
- *Oracle Database SecureFiles and Large Objects Developer's Guide*

Deprecated Features

The following feature is deprecated in this release, and may be desupported in a future release:

- Oracle Multimedia

There is no replacement for Oracle Multimedia. Oracle recommends that you replace Oracle Multimedia by using Oracle SecureFiles with third-party products for metadata extraction and image processing and conversion.

 **See Also:**

Oracle Database SecureFiles and Large Objects Developer's Guide

The following features were deprecated in Oracle Database 12c Release 2 (12.2):

- Oracle Multimedia support for the SQL/MM Still Image standard
- Oracle Multimedia Java client proxy classes, including DICOM proxy classes, and Oracle Multimedia servlet and JSP classes
- Oracle Multimedia support for DICOM

The following feature was deprecated in Oracle Database 12c Release 1 (12.1), and may be desupported in a future release:

- The Oracle Multimedia DICOM support in ORDImage objects that was introduced in Oracle Database 10g Release 2 (10.2)

New Features

The following feature was new in Oracle Database 12c Release 2 (12.2):

- Oracle Multimedia PL/SQL API

Oracle Multimedia now provides a simplified PL/SQL API with functions and procedures for managing image, audio, and video media data stored in BLOBs and BFILES in Oracle Database. This API allows metadata extraction and image processing operations to be more intuitively included in SQL and PL/SQL applications.

The PL/SQL API enables developers to include common operations in their applications. Examples include creating thumbnail images, cropping images, and converting images to Web-friendly formats, as well as extracting metadata. Using this API along with a comprehensive set of database tools and features, developers can consolidate multimedia data with other types of data in the database for easy display in reports and Web user interfaces. These advantages enable rapid development and deployment of all database applications that include multimedia data.

See *Oracle Multimedia Reference* for more information about the Oracle Multimedia PL/SQL Packages.

1

Introduction to Oracle Multimedia

This chapter provides an overview of Oracle Multimedia.

Oracle Multimedia (formerly Oracle interMedia) enables Oracle Database to store, manage, and retrieve images, audio, video, or other heterogeneous media data in an integrated fashion with other enterprise information.

Oracle Multimedia extends Oracle Database reliability, availability, and data management to multimedia content in traditional, medical, Internet, electronic commerce, and media-rich applications. Oracle Multimedia does not control media capture or output devices; this function is left to application software.

Oracle Multimedia provides these services and support:

- Image services for the storage, retrieval, metadata extraction, and processing of two-dimensional, static, bit-mapped images. Images are stored efficiently using popular compression schemes in industry-standard image formats for desktop publishing.
- Digital Imaging and Communications in Medicine (DICOM) support for the storage, retrieval, metadata extraction, processing, writing, conformance validation, and making anonymous of medical images and other DICOM content.

 **Note:**

DICOM support was deprecated in Oracle Database 12c Release 2 (12.2), and may be desupported in a future release.

- Audio and video services for the storage, retrieval, and metadata extraction of popular audio and video file formats.

This chapter includes these sections:

- [Oracle Multimedia Architecture](#)
- [Object Relational Technology](#)
- [Oracle Multimedia Capabilities](#)
- [Audio Concepts](#)
- [ORDDoc or Heterogeneous Media Data Concepts](#)
- [Image Concepts](#)
- [Video Concepts](#)
- [Loading Multimedia Data](#)
- [Multimedia Storage and Querying](#)
- [Accessing Multimedia Data](#)

 **See Also:**

- *Oracle Multimedia Reference* for detailed information about Oracle Multimedia APIs and their components
- *Oracle Multimedia DICOM Developer's Guide* for more information about Oracle Multimedia DICOM support

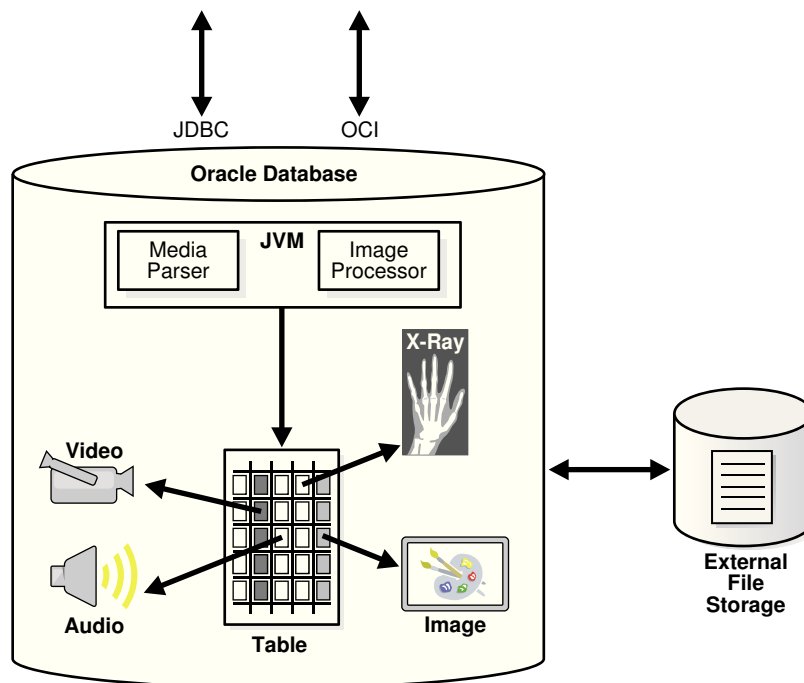
1.1 Oracle Multimedia Architecture

Oracle Multimedia is a single, integrated feature that extends the database by storing, managing, and retrieving image, audio, and video data.

The Oracle Multimedia architecture defines the framework through which media-rich content is supported in the database, along with traditional data. This content can then be securely shared across multiple applications written with popular languages and tools, easily managed and administered by relational database management and administration technologies, and offered on a scalable database that supports thousands of users.

The following figure illustrates the Oracle Multimedia architecture from the database perspective.

Figure 1-1 Oracle Multimedia Architecture



Using Oracle Multimedia, Oracle Database holds rich content in tables along with traditional data. As illustrated, rich content includes audio, image, video, and DICOM content, such as X-rays. Through a database-embedded JVM, a server-side media

parser and an image processor are supported. The media parser supports format and application metadata parsing, and can be extended to support additional formats. The image processor provides image processing for operations such as producing thumbnail-size images, converting image formats, and image watermarking.

Using Oracle Multimedia procedures or methods enables import and export operations between the database and external file storage systems. The double-sided arrow connecting Oracle Database with External File Storage shows this data movement.

Using Oracle Multimedia enables Oracle Database to store, manage, process, and retrieve DICOM content in database tables. DICOM content includes single-frame and multiframe images, waveforms, slices of 3-D volumes, video segments, and structured reports.

See Also:

- *Oracle Multimedia DICOM Developer's Guide* for a view and description of the complete architecture for Oracle Multimedia DICOM

1.2 Object Relational Technology

Oracle Database is an object relational database management system that provides support for multimedia data stored in BLOBs and BFILEs as well as in Oracle Multimedia object types.

In addition to its traditional role in the safe and efficient management of relational data, Oracle Database provides support for the safe and efficient storage and management of large objects (LOBs) using SecureFiles LOBs.

Oracle Multimedia provides four PL/SQL packages, which contain functions and procedures for managing image, audio, and video data stored in binary large objects (BLOBs) and external large objects (BFILEs) in Oracle Database:

- ORD_AUDIO for audio data
- ORD_DOC for heterogeneous data
- ORD_IMAGE for image data
- ORD_VIDEO for video data

In addition, Oracle Multimedia provides the ORD_DICOM PL/SQL package to support DICOM content produced by medical devices.

Developers can use the PL/SQL packages to include common operations, such as these, in their applications:

- Creating thumbnail images
- Cropping images
- Converting images into Web-friendly formats
- Extracting information from multimedia data, either as an XML string or as XML and individual attributes
- Loading multimedia data from operating system files into Oracle Database

- Exporting multimedia data from Oracle Database into operating system files

Oracle Database also supports the definition of object types, including the data associated with objects and the operations (methods) that can be performed on them. Complex objects include: digitized audio, image, video, and Digital Imaging and Communications in Medicine (DICOM) format medical images and other data.

Oracle Multimedia provides four object relational types, which store data source information in an object relational type known as ORDSource:

- ORDAudio, for audio data characteristics
- ORDDoc, for heterogeneous data characteristics
- ORDImage, for image data characteristics
- ORDVideo, for video data characteristics

In addition, Oracle Multimedia provides the ORDDicom object relational type, for characteristics of DICOM content produced by medical devices.

See Also:

- *Oracle Database SecureFiles and Large Objects Developer's Guide* for detailed information about using BLOBs and BFILEs
- *Oracle Multimedia Reference* for reference information about the PL/SQL packages and object types and methods for audio, heterogeneous, image, and video media, and for more information about the ORDSource object type and methods
- *Oracle Multimedia DICOM Developer's Guide* for reference and other information about the ORD_DICOM PL/SQL package and the ORDDicom object type and methods for DICOM format medical images and other data

1.3 Oracle Multimedia Capabilities

The capabilities of Oracle Multimedia include the storage, retrieval, management, and manipulation of multimedia data managed by Oracle Database.

Multimedia applications have common and unique requirements. Oracle Multimedia supports common application requirements and can be extended to address application-specific requirements. With Oracle Multimedia, multimedia data can be managed as easily as standard attribute data.

Database applications written in Java, C++, or traditional third-generation languages (3GLs) can interact with Oracle Multimedia through modern class library interfaces, or PL/SQL and Oracle Call Interface (OCI).

Oracle Multimedia supports storage of the popular file formats, including desktop publishing images, and streaming audio and video formats in databases. Oracle Multimedia provides the means to add audio, image, and video, or other heterogeneous media columns or objects to existing tables, and insert and retrieve multimedia data. This support enables database designers to extend existing databases with multimedia data, or to build new end-user multimedia database

applications. Oracle Multimedia developers can use the basic functions provided here to build specialized multimedia applications.

Oracle Multimedia defines PL/SQL packages and object types, similar to Java or C++ classes, to describe and process multimedia data. The PL/SQL packages are named `ORD_AUDIO`, `ORD_DOC`, `ORD_IMAGE`, and `ORD_VIDEO`. The object types are called `ORDAudio`, `ORDDoc`, `ORDImage`, and `ORDVideo`. Users who wish to store media data directly in BLOBs or BFILEs can use the Oracle Multimedia PL/SQL packages. Users who prefer media data and attributes to be encapsulated in a single object type can use the Oracle Multimedia object types.

The Oracle Multimedia PL/SQL packages contain functions and procedures for managing media data, including image, audio, and video data. The PL/SQL packages operate on media data stored in BLOBs and BFILEs. BLOBs store media data in the database under transaction control. Under transaction control, BFILEs store a pointer to the media data stored in an external file that is not under transaction control. The PL/SQL packages include procedures to extract metadata from media data. Metadata is information about the media data, such as object length, compression type, or format. Procedures are also provided to perform operations on the media data, such as `getProperties()` and `scale()`.

An instance of the Oracle Multimedia object types consists of attributes, including [metadata](#) and the [media data](#), and [methods](#). Media data is the actual audio, image, or video, or other heterogeneous media data. Metadata is information about the data, such as object length, compression type, or format. Methods are procedures that can be performed on objects, such as `getContent()` and `setProperties()`.

The Oracle Multimedia object types have a common media data storage model. The media data component of these objects can be stored in the database, in a BLOB under transaction control. The media data can also be stored outside the database, without transaction control. In this case, a pointer is stored in the database under transaction control, and the media data is stored in:

- File-based large object (BFILE)
- An HTTP server-based URL
- A user-defined source on a specialized media data server, or other server

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

When using Oracle Multimedia object types, media metadata is stored in the database under transaction control. Whether media data is stored within or outside the database, Oracle Multimedia manages metadata for all the media object types and might automatically extract it for audio, image, and video. When storing media data in BLOBs or BFILEs and using the Oracle Multimedia PL/SQL packages, it is up to the application to decide how to manage metadata.

Media metadata includes these attributes:

- Storage information about audio, image, and video, or other heterogeneous media data, including the source type, location, and source name, and whether the data is stored locally (in the database) or externally
- Update time stamp information for audio, image, and video, or other heterogeneous media data

- Audio and video data description
- Audio, image, and video, or other heterogeneous media data format
- MIME type of the audio, image, and video, or other heterogeneous media data
- Audio characteristics: encoding type, number of channels, sampling rate, sample size, compression type, and play time (duration)
- Image characteristics: height and width, image content length, image content format, and image compression format
- Video characteristics: frame width and height, frame resolution, frame rate, play time (duration), number of frames, compression type, number of colors, and bit rate
- Extracted metadata in XML, such as the director or producer of a movie

In addition to metadata extraction procedures and object methods, a set of image manipulation procedures and object methods is provided. For images, this includes format conversion, page selection, quantize operations, compression, scaling, cropping, copying, flipping, mirroring, rotating, sharpening, adjusting the gamma (brightness), adding watermarks to images, removing metadata from images, and embedding metadata into images.

Oracle Multimedia is a building block for various multimedia applications, rather than an end-user application. It consists of PL/SQL package procedures, and object types and their respective methods, for managing and processing multimedia data. Some example applications for Oracle Multimedia are:

- Repositories for digital check images
- Electronic health records, including DICOM medical images
- Call centers (for example, 911 and product call centers)
- Physical asset inventories
- Distance learning and online learning
- Real estate marketing
- Stock photography archives (for example, digital art galleries and professional photographers)
- Document imaging archives
- Financial news service customer information
- Web publishing
- Audio and video Web stores

Related Topics

- [Loading Multimedia Data](#)
Multimedia data can be managed best by Oracle Database. Load your multimedia data into the database to take advantage of its reliability, scalability, availability, and data management capabilities.

 **See Also:**

Oracle Multimedia Reference for more information about image processing operations

1.3.1 Oracle Multimedia Support for CDBs

Oracle Database 12c Release 1 (12.1) introduced multitenant container databases (CDBs).

A CDB is a single, physical database that can contain zero, one, or many customer-created pluggable databases (PDBs). A PDB is a portable collection of schemas, schema objects, and nonschema objects that appears to an Oracle Net client as a non-CDB. A non-CDB is a traditional Oracle database that cannot contain PDBs. Oracle Multimedia is supported in both CDB and non-CDB architectures.

 **See Also:**

- *Oracle Database Concepts* for information about the multitenant architecture
- *Oracle Database Administrator's Guide* for information about managing CDBs and PDBs

1.3.2 Data Guard Rolling Upgrade Support for Oracle Multimedia

During a rolling upgrade, you can run different releases of an Oracle database on the primary and logical standby databases while you upgrade them, one at a time, incurring minimal downtime on the primary database.

In addition to supporting media data stored in BLOBs and BFILEs, logical standby databases support these Oracle Multimedia data types during database rolling upgrades using Data Guard SQL Apply:

- ORDImage
- ORDSource
- ORDDicom
- ORDDataSource

 **See Also:**

- *Oracle Multimedia DICOM Developer's Guide* for information about how this feature impacts the DICOM data model
- *Oracle Database Upgrade Guide* for information about database rolling upgrades
- *Oracle Data Guard Concepts and Administration* for information about these and other Oracle Data Guard features

1.4 Audio Concepts

This section contains information about digitized audio concepts, and information about using the ORD_AUDIO PL/SQL package and the ORDAudio object type to build audio applications.

Topics include:

- [Digitized Audio](#)
- [Audio Components](#)

1.4.1 Digitized Audio

Using the ORD_AUDIO PL/SQL package or the ORDAudio object type, audio data can be stored, retrieved, and managed in a database.

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

1.4.2 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 on, 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. Oracle Multimedia can store and retrieve audio data of any supported data format. Oracle Multimedia can automatically extract metadata from audio data of a variety of popular audio formats. Oracle Multimedia can also extract application attributes in XML form.

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.

 **See Also:**

Oracle Multimedia Reference for a list of supported data formats from which ORDAudio can extract and store attributes and other audio features

1.5 ORDDoc or Heterogeneous Media Data Concepts

This section contains information about heterogeneous media data concepts, and information about using the ORD_DOC PL/SQL package and the ORDDoc object type to build applications.

Topics include:

- [Digitized Heterogeneous Media Data](#)
- [Heterogeneous Media Data Components](#)

1.5.1 Digitized Heterogeneous Media Data

Oracle Multimedia integrates the storage, retrieval, and management of heterogeneous media data in a database.

Using the ORD_DOC PL/SQL package or the ORDDoc object type, any heterogeneous media data including audio, image, and video, can be stored in a database column. Instead of having separate columns for audio, image, text, and video objects, you can use one column of type BLOB, BFILE, or ORDDoc to represent all types of multimedia.

1.5.2 Heterogeneous Media Data Components

Heterogeneous media data components consist of the data (digitized bits) and attributes that describe and characterize the heterogeneous media data.

Heterogeneous media data can have different formats, depending upon the application generating the media data. Oracle Multimedia can store and retrieve media data of any supported data format. The BLOB, BFILE, and ORDDoc types can be used in applications that require you to store different types of heterogeneous media data (such as audio, image, video, and any other type of media data) in the same column so you can build a common metadata index on all the different types of media data. Using this index, you can search across all the different types of heterogeneous media data. You cannot use this same search technique if the different types of heterogeneous media data are stored in different types of objects, in different columns of relational tables.

ORDDoc can automatically extract metadata from data of a variety of popular audio, image, and video data formats. ORDDoc can also extract application attributes and store them in the comments attribute of the object in XML form. ORDDoc is extensible and can be made to recognize and support other heterogeneous media data formats.

 **See Also:**

- *Oracle Multimedia Reference* for a list of supported audio data formats from which ORDDoc can extract and store attributes
- *Oracle Multimedia Reference* for a list of supported image data formats from which ORDDoc can extract and store attributes
- *Oracle Multimedia Reference* for a list of supported video data formats from which ORDDoc can extract and store attributes

1.6 Image Concepts

This section contains information about digitized image concepts, and information about using the ORD_IMAGE PL/SQL package and the ORDImage object type to build image applications.

Topics include:

- [Digitized Images](#)
- [Image Components](#)
- [Metadata in Images](#)
- [Medical Imaging](#)
- [Metadata Extraction](#)
- [Image Processing](#)

1.6.1 Digitized Images

Using the ORD_IMAGE PL/SQL package or the ORDImage object type, digitized images can be stored, retrieved, and managed in a database.

Oracle Multimedia 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 digital 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.

1.6.2 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 the name of the person pictured in a photograph, description of the image, date photographed, photographer, and so on, 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. `ORDImage` can store and retrieve image data of any data format. `ORDImage` can process and automatically extract properties of images of a variety of popular data formats. In addition, certain foreign images (formats not natively supported by `ORDImage`) 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. As compared with [lossless compression schemes](#), [lossy compression schemes](#) generally provide higher compression.

The [image interchange format](#) describes a well-defined organization and use of image attributes, data, and often compression schemes, enabling different applications to create, exchange, and use images. Interchange formats are often stored as disk files. They can also be exchanged in a sequential fashion over a network and be referred to as [protocols](#). There are many application subdomains within the digitized imaging world and many applications that create or use digitized images within these. `ORDImage` supports storage and retrieval of all image data formats, and processing and attribute extraction of many image data formats.

See Also:

Oracle Multimedia Reference for a list of supported data formats from which Oracle Multimedia can process, extract, and store attributes and other image features

1.6.3 Metadata in Images

Oracle Database provides an image metadata feature in Oracle Multimedia. Metadata can be stored in a database, indexed, searched, and made available to applications using the standard mechanisms of Oracle Database.

The image metadata feature adds the ability to read (or extract) and write (or embed) application metadata in images. In addition, this feature adopts a standard way to represent metadata when it is separate from an image file.

Related Topics

- [Working with Metadata in Oracle Multimedia Images](#)
Image files can contain information about the content of the images, the image pixel data, and image metadata.

1.6.4 Medical Imaging

Oracle Database includes medical imaging format support in Oracle Multimedia DICOM.

 **Note:**

The Oracle Multimedia support for DICOM was deprecated in Oracle Database 12c Release 2 (12.2), and may be desupported in a future release. Oracle recommends that you stop using deprecated features as soon as possible.

Medical imaging format support comprises these Oracle Multimedia DICOM features:

- Storage and retrieval of medical imaging data in the database to synchronize the DICOM data with the associated business data
- Full PL/SQL and object interfaces to Oracle Multimedia DICOM services
- Extraction of DICOM metadata according to user-specifiable XML documents
- Querying using associated relational data and extracted metadata
- Image processing, such as thumbnail generation
- Creation of new DICOM objects
- Conformance validation based on a set of user-specified conformance rules
- Making DICOM objects anonymous based on user-defined rules that specify the set of attributes to be made anonymous and how to make those attributes anonymous
- The ability to update run-time behaviors, such as the version of the DICOM standard supported, without installing a new release of Oracle Database

 **See Also:**

Oracle Multimedia DICOM Developer's Guide for more information about Oracle Multimedia DICOM

1.6.5 Metadata Extraction

Oracle Multimedia provides the ability to extract format metadata from media sources.

Once metadata has been extracted and stored, you can index the metadata to allow media queries based on metadata.

 **See Also:**

The `setProperty()` method in *Oracle Multimedia Reference* for more information about metadata extraction

1.6.6 Image Processing

Oracle Multimedia supports several types of image processing.

Oracle Multimedia image processing support includes format transcoding, cutting, scaling, generating thumbnail images, and applying watermarks. In addition, when the destination image file format is RAW Pixel (RPIX) or Microsoft Windows Bitmap (BMPF), Oracle Multimedia supports several operators for changing the format characteristics.



See Also:

Oracle Multimedia Reference for more information about image processing

1.7 Video Concepts

This section contains information about digitized video concepts, and information about using the ORD_VIDEO PL/SQL package and the ORDVideo object type to build video applications.

Topics include:

- [Digitized Video](#)
- [Video Components](#)

1.7.1 Digitized Video

Using the ORD_VIDEO PL/SQL package or the ORDVideo object type, video data can be stored, retrieved, and managed in a database.

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

1.7.2 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 on, within the video data.

The video data can have different formats, compression types, frame rates, frame sizes, frame resolutions, playing times, compression types, number of colors, and bit rates depending upon how the video data was digitally recorded. Oracle Multimedia can:

- Automatically extract metadata from video data of a variety of popular video formats
- Extract application attributes and store them in the comments attribute of the object in XML form

- Be made to recognize and support additional video formats (because it is extensible)

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.

 **See Also:**

Oracle Multimedia Reference for a list of supported data formats from which Oracle Multimedia can extract and store attributes and other video features

1.8 Loading Multimedia Data

Multimedia data can be managed best by Oracle Database. Load your multimedia data into the database to take advantage of its reliability, scalability, availability, and data management capabilities.

To bulk load multimedia data into the database, you can use:

- **SQL*Loader**
SQL*Loader is an Oracle utility that lets you load data, and in this case, multimedia data (LOB data), from external multimedia files into a table of a database containing BLOB or Oracle Multimedia object type columns.
- **PL/SQL**
A procedural extension to SQL, PL/SQL is an advanced fourth-generation programming language (4GL) of Oracle. You can write PL/SQL procedures to load multimedia data from BLOB, file system, and URL media data sources into BLOB or Oracle Multimedia object type columns.

An advantage of using SQL*Loader is that it is easy to create and test the control file that controls your data loading operation.

An advantage of using PL/SQL scripts to load your data is that you can call procedures or methods as you load data to generate thumbnail images, or extract properties.

 **See Also:**

- *Oracle Database Utilities* for more information about SQL*Loader
- *Oracle Database PL/SQL Language Reference* for more information about PL/SQL procedures

1.9 Multimedia Storage and Querying

Media data can be stored directly in BLOBs or BFILEs, or in Oracle Multimedia object types.

The features of Oracle Multimedia are available to media stored in BLOBs and BFILES using the Oracle Multimedia PL/SQL API. This PL/SQL API lets developers do the following with media data stored in BLOBs and BFILES: move data between the local file system and the database; parse and extract the properties of the media data; and store these properties in an XMLType or an XML formatted CLOB, and optionally, in individual relational columns. Developers are not required to make changes to their existing application schema or to instantiate Oracle Multimedia object types to take advantage of the PL/SQL API. The Oracle Multimedia PL/SQL API can also be used to perform image processing operations such as cut, scale, compress, and convert format.

The ORDAudio, ORDDoc, ORDImage, and ORDVideo object types all contain an attribute of type ORDSrc and methods for multimedia data source manipulation.

 **Note:**

Do not call ORDSrc methods directly. Instead, invoke the wrapper method of the media object corresponding to the ORDSrc method. This information is presented for users who want to write their own user-defined sources.

The following subsections briefly describe storage and querying:

- [Storing Multimedia Data](#)
- [Querying Multimedia Data](#)

 **See Also:**

Oracle Multimedia Reference for reference information about the Oracle Multimedia PL/SQL API and the Oracle Multimedia object types and methods for audio, heterogeneous, image, and video media, and for more information about the ORDSrc object type and methods

1.9.1 Storing Multimedia Data

Oracle Multimedia can store multimedia data as an internal source within the database, under transactional control as a BLOB. It can also externally reference digitized multimedia data stored as an external source in an operating system-specific file in a local file system, as a URL on an HTTP server, or as a user-defined source on other servers, such as media servers. Although these external storage mechanisms are particularly convenient for integrating existing sets of multimedia data with a database, the multimedia data is not under transactional control if it is not stored in the database.

BLOBs are stored in the database tablespaces in a way that optimizes space and provides efficient access. Large BLOBs cannot be stored inline (BLOBs under 4 kilobytes can be stored inline) with other row data. Depending on the size of the BLOB, a locator is stored in the row and the actual BLOB (up to 8 terabytes to 128 terabytes, depending on the block size) is stored in other tablespaces. The locator can be considered a pointer to the actual location of the BLOB value. When you select a BLOB, you are selecting the locator instead of the value, although this is done

transparently. An advantage of this design is that multiple BLOB locators can exist in a single row. For example, you might want to store a short video clip of a training tape, an audio recording containing a brief description of its contents, a syllabus of the course, a picture of the instructor, and a set of maps and directions to each training center all in the same row.

Because BFILEs are not under the transactional control of the database, users could change the external source without updating the database, thus causing an inconsistency with the BFILE locator.

Oracle Multimedia ORDAudio, ORDDoc, ORDImage, and ORDVideo object types provide wrapper methods over BLOBs and BFILEs to perform these source related functions:

- Set the source of the data as local or external
- Modify the time an object was last updated
- Set information about the external source type, location, and name of the data
- Transfer data into or out of the database
- Obtain information about the local data content such as its length, location, or its handle to the BLOB, put the content into a temporary BLOB, or delete it
- Access source data by opening it, reading it, writing to it, trimming it, and closing it

See Also:

- *Oracle Database SecureFiles and Large Objects Developer's Guide* for more information about BLOBs and BFILEs
- *Oracle Call Interface Programmer's Guide* for more information about BLOB and BFILE operations

1.9.2 Querying Multimedia Data

Once stored within a database, multimedia data can be queried and retrieved by using the various alphanumeric columns or object attributes of the table to find a row that contains the desired data. For example, you can select a video clip from the Training table where the course name is 'Oracle Database Concepts'.

Multimedia data can be queried by extracted metadata and by other relational table columns.

1.10 Accessing Multimedia Data

Applications access and manipulate multimedia data using SQL, PL/SQL, OCI, or Java.

SQL, PL/SQL, and OCI applications can manipulate and modify multimedia data by accessing the PL/SQL packages ORD_AUDIO, ORD_DOC, ORD_IMAGE, and ORD_VIDEO, or through the object relational types ORDAudio, ORDDoc, ORDImage, and ORDVideo

Java applications on any tier (client, application server, or database) can access, manipulate, and modify audio, image, and video data, or heterogeneous media data stored in a database by using anonymous PL/SQL code blocks that access the Oracle Multimedia PL/SQL packages or object relational types.

2

Oracle Multimedia Application Development

Oracle Multimedia enables you to develop either traditional client/server or two-tier applications, or multitier applications. Either method can then deploy Web applications to run on an application server tier, be tightly integrated with Oracle Database, and enable users to access the application from their desktop through a Web browser.

You can create production quality Oracle Multimedia applications for use in a production environment where users can interact with the application through either the standalone client interface or a Web browser. For Web applications, which are based on standards such as TCP/IP, HTTP, HTML, XML, and XHTML, this capability is facilitated by rapid developments in the underlying technology. As key software components become more tightly integrated, developers' tasks to design, create, and manage Web applications become faster, easier, and simpler to implement.

Several tools and languages can be used for application development with multimedia data. This chapter describes them in these sections:

- [Developing Multimedia Applications Using SQL Developer](#)
- [Developing Multimedia Applications Using Application Express](#)
- [Developing Multimedia Applications Using Java and JDBC](#)
- [Developing Multimedia Applications Using PL/SQL](#)
- [Developing PL/SQL Web Applications](#)

See Also:

- [Oracle Multimedia PL/SQL Photo Album Sample Application](#) for a description of the Oracle Multimedia Photo Album sample Web application, which uses PL/SQL scripts to demonstrate how to develop Web applications to upload and retrieve media data stored in a database
- [Oracle Multimedia Code Wizard Sample Application for the PL/SQL Gateway](#) for a description of the Oracle Multimedia Code Wizard application, which lets you create PL/SQL stored procedures for the PL/SQL Gateway for uploading and retrieving media data stored in a database using Oracle Multimedia object types

2.1 Developing Multimedia Applications Using SQL Developer

Oracle SQL Developer is a tool that allows standalone graphical browsing and development of database schema objects.

Oracle SQL Developer supports the BLOB and BFILE data types. Developers who store their multimedia data in BLOBs and BFILES can use Oracle SQL Developer for their multimedia applications.

Follow these steps to view an image that is stored in the database as a BLOB:

1. Find the table you created in the list of tables in SQL Developer.
2. Create a new row for importing images into your table. Then, import one or more images into the table.
3. Go to the Query Result area to view the properties of any imported image.
4. Select the BLOB in the column for the image you want to view. Then, double click the left mouse button.

A pencil icon displays to the right of the BLOB you selected.

5. Click the pencil icon to open the View Value window.
6. In the `View As:` field, check `Image`.

The image you selected is displayed on the Saved Image tab.

See Also:

- *Oracle SQL Developer User's Guide* for more information about using SQL Developer with BLOBs and BFILES and with multimedia data
- *Oracle Multimedia* on the Oracle Technology Network Web site for additional examples and other resources

2.2 Developing Multimedia Applications Using Application Express

Oracle Application Express is a rapid development tool for Web applications on Oracle Database.

Built-in features such as user interface themes, navigational controls, form handlers, and flexible reports, enable users of Oracle Application Express to accelerate the application development process.

Oracle Application Express provides support for BLOB and BFILE data types. Developers who store their multimedia data in BLOBs or BFILES can use Oracle Application Express for rapid application development.

 **See Also:**

Oracle Application Express App Builder User's Guide for information about building multimedia applications with Oracle Application Express

2.3 Developing Multimedia Applications Using Java and JDBC

Using the Java database connectivity (JDBC) interface, Oracle Multimedia enables users to embed anonymous PL/SQL code blocks to quickly develop Java applications for use on any tier (client, application server, or database) to manipulate and modify audio, image, and video data, or heterogeneous media data stored in a database.

The following subsections describe how to use Java and JDBC to develop multimedia applications:

- [Media Upload in Java](#)
- [Retrieval of Image Properties in Java](#)
- [Thumbnail Image Creation in Java](#)
- [Handling Oracle Multimedia Exceptions in Java](#)

The examples in this section assume a table, with these two columns: a numeric identifier (`id`), and a binary large object (BLOB) to hold the image itself (`image_blob`).

```
create table image_blob_table ( id number primary key, image_blob BLOB)  
lob(image_blob) store as securefile;
```

 **See Also:**

- *Oracle Database JDBC Developer's Guide* for more information about using JDBC
- *Oracle Multimedia Reference* for more information about the Oracle Multimedia PL/SQL packages and object interfaces
- *Oracle Multimedia DICOM Developer's Guide* for more information about Oracle Multimedia DICOM features and enhancements

2.3.1 Media Upload in Java

This topic uses an example to show how to upload multimedia data in Java.

The following example shows how to import multimedia data from the file system into the database in a Java application.

```
public void writeImageToDatabase(int id, String fileName )  
    throws SQLException, IOException  
    {  
    //Define the PL/SQL block to insert the new row.
```

```

final String INSERT_BLOB = "DECLARE "
    + "    src_id          NUMBER; "
    + "BEGIN "
    + "    src_id := ?;"
    + "    DELETE FROM image_blob_table WHERE id=src_id; "
    + "    INSERT INTO image_blob_table t (id, image_blob) "
    + "        VALUES(src_id, empty_blob()) "
    + "        RETURNING t.image_blob INTO ?; "
    + "END;";

try {

    //Create the statement object.
    final OracleCallableStatement pstmt =
        (OracleCallableStatement)connection.prepareCall
        (INSERT_BLOB);

    //Binding the variables to the statement.
    pstmt.setInt(1, id); //ID
    pstmt.registerOutParameter(2, OracleTypes.BLOB);
    pstmt.execute(); //Execute the PL/SQL statement.

    //Get the BLOB locator from the table.
    BLOB blob = pstmt.getBLOB(2);
    File binaryFile = new File(fileName);
    FileInputStream instream = new FileInputStream(binaryFile);

    //Retrieve the ideal buffer size to use in writing to the BLOB.
    int size = 1024*1024; // 1MB.
    byte[] buffer = new byte[size];
    int read = -1;
    long position =1;

    //Read the file to the byte array buffer, then write it to the BLOB.
    while ((read = instream.read(buffer)) != -1)

        {

            blob.setBytes(position,buffer,0,read);
            position+=read;

        }

    instream.close();
    connection.commit();

} catch (FileNotFoundException e) {
    throw new FileNotFoundException("File " + fileName + " not Found.");
} catch (IOException e) {
    throw new IOException("Error while reading " + fileName);
}

}
}

```

Call `writelnImageToDatabase()` passing the row id and source file path:

```

//Write data from a local file into a BLOB in the database.
quickstart.writelnImageToDatabase(1, "flowers.jpg");

```

 **Note:**

If the `autoCommit` flag on the connection is set to true, or is not set (the default is true), the following error is returned when you attempt to select a row with a BLOB column for update:

```
java.sql.SQLException: ORA-22990: LOB locators cannot span transactions
```

You can set the `autoCommit` flag to false as follows:

```
conn.setAutoCommit(false);
```

2.3.2 Retrieval of Image Properties in Java

This topic uses an example to show how to retrieve the properties of image data in Java.

After the image data is imported from the file system into the table `image_blob_table`, the database does not know what the binary bytes in the BLOB column `image_blob` represent. The following example shows how to use the `ORDSYS.ORD_IMAGE.getProperties()` procedure of the Oracle Multimedia PL/SQL package to extract the image properties into the Java application.

```
public HashMap <String, Object> getProperties_example_j(int id )
    throws SQLException
{
    //Define the PL/SQL block to extract the properties.
    final String getPropertiesStmt = "DECLARE "
        + "    src                                BLOB; "
        + "    img_mimeType                        VARCHAR2(32); "
        + "    img_width                            INTEGER; "
        + "    img_height                            INTEGER; "
        + "    img_contentLength                    INTEGER; "
        + "    img_fileFormat                        VARCHAR2(32); "
        + "    img_contentFormat                    VARCHAR2(32); "
        + "    img_compressionFormat                VARCHAR2(32); "
        + "BEGIN "
        + "    SELECT image_blob INTO src FROM image_blob_table"
        + "        WHERE id=?; "
        + "    ORDSYS.ORD_IMAGE.getProperties(src, "
        + "        img_mimeType, "
        + "        img_width, "
        + "        img_height, "
        + "        img_fileFormat, "
        + "        img_compressionFormat, "
        + "        img_contentFormat, "
        + "        img_contentLength); "
        + "    ? := img_mimeType; "
        + "    ? := img_width; "
        + "    ? := img_height; "
        + "    ? := img_contentLength; "
        + "    ? := img_fileFormat; "
        + "    ? := img_contentFormat;"
        + "    ? := img_compressionFormat; "
        + "END;";

    //Create the statement object.
```

```

final OracleCallableStatement pstmt =
    (OracleCallableStatement)connection.prepareCall(getPropertiesStmt);

//Binding the variables to the statement.
pstmt.setInt(1, id);
pstmt.registerOutParameter(2, OracleTypes.VARCHAR);
pstmt.registerOutParameter(3, OracleTypes.INTEGER);
pstmt.registerOutParameter(4, OracleTypes.INTEGER);
pstmt.registerOutParameter(5, OracleTypes.INTEGER);
pstmt.registerOutParameter(6, OracleTypes.VARCHAR);
pstmt.registerOutParameter(7, OracleTypes.VARCHAR);
pstmt.registerOutParameter(8, OracleTypes.VARCHAR);

//Execute the statement.
pstmt.execute();

//Create a HashMap object and populate it with the properties.
HashMap<String, Object> map = new HashMap<String, Object>();
map.put("mimeType", pstmt.getString(2));
map.put("width", pstmt.getInt(3) );
map.put("height", pstmt.getInt(4));
map.put("contentLength", pstmt.getInt(5));
map.put("fileFormat", pstmt.getString(6));
map.put("contentFormat", pstmt.getString(7));
map.put("compressionFormat", pstmt.getString(8));

return map;
}

```

Call `getProperties_example_j()` passing the id, then iterate over the `HashMap` to print the properties.

```

System.out.println("Original image properties");
HashMap<String, Object> attributesMap= quickstart.getProperties_example_j(1);
//Iterate over the HashMap.
for (Map.Entry<String, Object> entry : attributesMap.entrySet()) {
    System.out.println(entry.getKey() + " = " + entry.getValue());
}

```

Note:

If the image data that is in the `image_blob` column is not a supported format for Oracle Multimedia (for example: PSD), the following error is returned.

```

Exception in thread "main" java.sql.SQLException: ORA-29400: data
cartridge error

```

2.3.3 Thumbnail Image Creation in Java

This topic uses an example to show how to generate a thumbnail image from an existing image in Java.

The Oracle Multimedia `ORD_IMAGE` PL/SQL package includes several image processing operations that you can invoke within the database. To generate a thumbnail image from an existing image, you can use the

ORDSYS.ORD_IMAGE.thumbnail() procedure of the Oracle Multimedia PL/SQL package.

The following example shows how to create a thumbnail image from a source BLOB.

```
public void thumbnail_example_j(int src_id, int dst_id ) throws SQLException
{
    //Define the PL/SQL block to create a thumbnail.
    final String createThumbnailStmt = "DECLARE "
        + "    src_blob      BLOB;"
        + "    dst_blob      BLOB;"
        + "    src_id        NUMBER;"
        + "    dst_id        NUMBER;"
        + "BEGIN"
        + "    src_id := ?;"
        + "    dst_id := ?;"
        + "    DELETE FROM image_blob_table WHERE id = dst_id;"
        + "    INSERT INTO image_blob_table(id, image_blob) "
        + "        VALUES (dst_id, empty_blob()) "
        + "        RETURNING image_blob INTO dst_blob;"
        + "    SELECT image_blob INTO src_blob FROM image_blob_table"
        + "        WHERE id = src_id;"
        + "    ORDSYS.ORD_IMAGE.thumbnail(src_blob,dst_blob);"
        + "    UPDATE image_blob_table SET image_blob = dst_blob"
        + "        WHERE id = dst_id; "
        + "END;";

    final OracleCallableStatement pstmt =
        (OracleCallableStatement)connection.prepareCall(createThumbnailStmt);

    //Binding the variables to the statement.
    pstmt.setInt(1, src_id);
    pstmt.setInt(2, dst_id);
    //Execute the statement.
    pstmt.execute();
    connection.commit();
}
```

Call thumbnail_example_j() passing the source and destination ids.

```
//Create a thumbnail.
quickstart.thumbnail_example_j(1,2);
```

2.3.4 Handling Oracle Multimedia Exceptions in Java

This topic demonstrates proper error handling practices using code examples. These examples show how to handle some common Oracle Multimedia errors and other types of errors in Java programs.

Possible errors that can occur during run time should always be handled in your application. This practice enables the program to continue its operation even when it encounters a run-time error. This practice also enables users to know what went wrong during program operation. Proper error handling practices ensure that, whenever possible, you are always able to recover from an error while running an application. In addition, proper error handling provides you with the information you need so you always know what went wrong.

When handling exceptions, Java uses the try/catch block. For example, in Java, the exception can appear as follows:

```
try {
    //<some program logic>
}
catch (exceptionName a) {
    //Exception logic
}
finally {
    //Execute logic if try block is executed even if an exception is caught
}
```

When you design, code, and debug your application, you are aware of the places in your program where processing might stop due to a failure to anticipate an error. Those are the places in your program where you must add exception handling blocks to handle the potential errors.

The examples in this section describe exception handling using the try/catch block.

The following subsections provide additional details and examples of exception handling in Java:

- [Handling the Setting of Properties for Unknown Image Formats in Java](#)
- [Handling Image Processing for Unknown Image Formats in Java](#)

See Also:

- *Oracle Database Java Developer's Guide* for more information about handling Java exceptions
- *Oracle Database JDBC Developer's Guide* for more information about handling Java exceptions using JDBC

2.3.4.1 Handling the Setting of Properties for Unknown Image Formats in Java

This topic shows how to handle exceptions when setting properties for unknown image formats in Java.

The following try/catch block shows how to handle exceptions on the `setProperties()` method:

```
try
{
    img.setProperties();
    return true;
}
catch (SQLException e)
{
    return false;
}
```

If an exception is thrown, the `setProperties()` method returns `false` to indicate failure; otherwise it returns `true`.

2.3.4.2 Handling Image Processing for Unknown Image Formats in Java

This topic shows how to handle exceptions when processing images with unknown formats in Java.

If an application tries to process an image in cases when the image format is unknown, then when the application calls the `processCopy()` method, the application always fails. To work around this potential problem, the application uses the following try/catch block to catch any SQL exceptions:

```
try
{
    image.processCopy( "maxScale=50,50", thumb );
}
catch ( SQLException e )
{
    thumb.deleteContent();
    thumb.setContentLength( 0 );
}
```

In this example, when the image format is unknown and a thumbnail image cannot be created, the application catches the SQL exception and calls the `deleteContent()` method to delete the content of the thumbnail image, and then calls the `setContentLength()` method to set its length to zero.

2.4 Developing Multimedia Applications Using PL/SQL

Using the PL/SQL application programming interface, Oracle Multimedia enables users to develop PL/SQL applications to manipulate audio, image, and video data, or heterogeneous media data stored in a database.

PL/SQL is a completely portable, high-performance transaction processing language that combines the data manipulation power of SQL with the data processing power of procedural languages.

This topic briefly describes how to manipulate Oracle Multimedia database objects with the PL/SQL application programming interface (API). The following Oracle Multimedia object types are available for storing media in the database:

- ORDAudio
- ORDDicom
- ORDDoc
- ORDImage
- ORDVideo

Although this topic primarily discusses the Oracle Multimedia object types, the PL/SQL API can also be used to manipulate multimedia data stored directly in BLOBs or BFILEs. The following Oracle Multimedia PL/SQL packages are available for manipulating multimedia data in the database:

- ORD_AUDIO
- ORD_DICOM
- ORD_DOC

- `ORD_IMAGE`
- `ORD_VIDEO`

The examples in this section use the Oracle Database Sample Schemas, which are available on GitHub.

The following subsections describe how to develop multimedia applications using PL/SQL:

- [Setting Up Directory Objects for Media Import and Export](#)
- [Media Upload in PL/SQL](#)
- [Media Query in PL/SQL](#)
- [Media Download in PL/SQL](#)
- [Handling Oracle Multimedia Exceptions in PL/SQL](#)



See Also:

Oracle Multimedia Reference for details about the Oracle Multimedia PL/SQL packages and object types

2.4.1 Setting Up Directory Objects for Media Import and Export

This topic describes how to set up directory objects and grant privileges for reading and writing multimedia data to the file system using PL/SQL.

To access files with PL/SQL, you must create a directory object in the database that points to a directory that is accessible by the database server. For example, the following command creates the `MEDIA_DIR` directory in the sample schema:

```
CREATE DIRECTORY MEDIA_DIR AS  
  'c:\oracle\product\10.2.0\db_1\demo\schema\product_media';
```

To retrieve media data from the database to a file, you must grant the write permission on the specified directory to the appropriate user. For example:

```
GRANT WRITE ON DIRECTORY MEDIA_DIR TO SCOTT;
```

To upload media data from a file to the database, you must grant the read permission on the specified directory to the appropriate user. For example:

```
GRANT READ ON DIRECTORY MEDIA_DIR TO SCOTT;
```

2.4.2 Media Upload in PL/SQL

This topic uses an example to show how to upload multimedia data in PL/SQL.

Media upload means importing media data from the file system into the database tablespaces. The following series of steps is typical:

1. Insert a new row into the table, creating new objects by using the object constructor of the Oracle Multimedia object type.

2. Call the import method of the Oracle Multimedia object to bring the data from the file system into the database.
3. Call the setProperties method of the Oracle Multimedia object to determine and populate the attributes of the object.
4. Update the table so that the Oracle Multimedia object in the table contains the attribute values extracted in the previous step.

The PL/SQL code that implements these steps for inserting a new row in the PM.ONLINE_MEDIA table is shown in this example:

```
DECLARE
  img ORDIImage;
  aud ORDAudio;
  vid ORDVideo;
  ctx RAW(64) := NULL;
BEGIN
  -- Insert a new row into the pm.online_media table.
  DELETE FROM pm.online_media WHERE product_id = 3003;
  INSERT INTO pm.online_media
    (product_id,
     product_photo,
     product_audio,
     product_video)
  VALUES (3003,
          ORDSYS.ORDImage('FILE', 'MEDIA_DIR', 'laptop.jpg', 0),
          ORDSYS.ORDAudio('FILE', 'MEDIA_DIR', 'laptop.mpa', 0),
          ORDSYS.ORDVideo('FILE', 'MEDIA_DIR', 'laptop.rm', 0))
  RETURNING product_photo, product_audio, product_video
  INTO img, aud, vid;

  -- Bring the media into the database and populate the attributes.

  -- ORDIImage.import also calls ORDIImage.setProperties.
  img.import(ctx);

  aud.import(ctx);
  aud.setProperties(ctx);

  vid.import(ctx);
  vid.setProperties(ctx);

  -- Update the table with the properties we have extracted.
  UPDATE pm.online_media
  SET   product_photo = img,
        product_audio = aud,
        product_video = vid
  WHERE product_id = 3003;

  COMMIT;
END;
/
```

2.4.3 Media Query in PL/SQL

This topic uses examples to show how to query multimedia data in PL/SQL.

You can include media attributes (for example: height, width, and MIME type) in standard SQL queries by using accessor methods (for example: getHeight, getWidth, and getMimeType). [Example 2-1](#), [Example 2-2](#), and [Example 2-3](#) show how to use

these accessor methods to query one or more object attributes for image, audio, and video objects, respectively.

Example 2-1 Image Query (Height, Width, and MimeType Attributes)

```
SELECT t.product_id          id,
       t.product_photo.getHeight() height,
       t.product_photo.getWidth() width,
       t.product_photo.getMimeType() mimetype
FROM pm.online_media t;
```

Example 2-2 Audio Query (MimeType Attribute)

```
SELECT t.product_id          id,
       t.product_audio.getMimeType() mimetype
FROM pm.online_media t;
```

Example 2-3 Video Query (MimeType Attribute)

```
SELECT t.product_id          id,
       t.product_video.getMimeType() mimetype
FROM pm.online_media t;
```

2.4.4 Media Download in PL/SQL

This topic uses an example to show how to download multimedia data in PL/SQL.

To download media from the database into a file on the file system, call the export method of the Oracle Multimedia object. The following code example exports the image in the row with product_id 3117 to a file named 3117.jpg in the directory MEDIA_DIR. This code example highlights in bold the PL/SQL statements where this export operation takes place.

```
DECLARE
  img ORDImage;
  ctx RAW(64) := NULL;
BEGIN
  SELECT product_photo
  INTO img
  FROM pm.online_media
  WHERE product_id = 3117;
  img.export(ctx, 'FILE', 'MEDIA_DIR', '3117.jpg');
END;
```

2.4.5 Handling Oracle Multimedia Exceptions in PL/SQL

This topic demonstrates proper error handling practices using code examples. These examples show how to handle some common Oracle Multimedia errors and other types of errors in PL/SQL programs.

Possible errors that can occur during run time should always be handled in your application. This practice enables the program to continue its operation even when it encounters a run-time error. This practice also enables users to know what went wrong during program operation. Proper error handling practices ensure that, whenever possible, you are always able to recover from an error while running an application. In addition, proper error handling provides you with the information you need so you always know what went wrong.

When handling exceptions, PL/SQL uses exception blocks. For example, in PL/SQL, the exception can appear as follows:

```
BEGIN
<some program logic>
EXCEPTION
    WHEN OTHERS THEN
        <some exception logic>
END;
```

When you design, code, and debug your application, you are aware of the places in your program where processing might stop due to a failure to anticipate an error. Those are the places in your program where you must add exception handling blocks to handle the potential errors.

These examples are extracted from the PL/SQL sample applications that are described in [Oracle Multimedia PL/SQL Photo Album Sample Application](#) and [Oracle Multimedia Code Wizard Sample Application for the PL/SQL Gateway](#).

The following subsections provide additional details and examples of exception handling in PL/SQL:

- [Handling the Setting of Properties for Unknown Image Formats in PL/SQL](#)
- [Handling Image Processing for Unknown Image Formats in PL/SQL](#)

See Also:

Oracle Database PL/SQL Language Reference for more information about handling PL/SQL exceptions

2.4.5.1 Handling the Setting of Properties for Unknown Image Formats in PL/SQL

This topic shows how to handle exceptions when setting properties for unknown image formats in PL/SQL.

If your program tries to set the properties of an uploaded image (it reads the image data to get the values of the object attributes so it can store them in the appropriate attribute fields) and the image format is not recognized, then the `setProperties()` method fails. To catch this exception and work around this potential problem, the application uses the following exception block:

```
BEGIN
    new_image.setProperties();
EXCEPTION
    WHEN OTHERS THEN
        new_image.contentLength := upload_size;
        new_image.mimeType := upload_mime_type;
END;
```

In this example, this exception handler sets the MIME type and length of the image based on the values from the upload table described at the beginning of the `insert_new_photo` procedure. The browser sets a MIME type header when the file is uploaded. The application reads this header to set the `ORDImage` field.

2.4.5.2 Handling Image Processing for Unknown Image Formats in PL/SQL

This topic shows how to handle exceptions when setting properties for unknown image formats in PL/SQL.

If your program tries to process an image in cases when the image format is unknown, the `processCopy()` method always fails. To work around this potential problem, the application uses the following exception block:

```
BEGIN
    new_image.processCopy( 'maxScale=50,50', new_thumb);
EXCEPTION
    WHEN OTHERS THEN
        new_thumb.deleteContent();
        new_thumb.contentLength := 0;
END;
```

In this example from the Oracle Multimedia PL/SQL Web Toolkit Photo Album application, when the image format is unknown and a thumbnail image cannot be created, this exception handler deletes the content of the thumbnail image and sets its length to zero.

2.5 Developing PL/SQL Web Applications

You can use the PL/SQL Gateway and PL/SQL Web Toolkit to develop Web applications in PL/SQL.



Note:

The use of the PL/SQL Gateway and PL/SQL Web Toolkit is suitable for applications that require tight control of the HTTP communication and HTML generation. For other applications, consider using Oracle Application Express, which provides more features and a convenient graphical interface to ease application development.

Developing Web applications using PL/SQL consists of developing one or more PL/SQL packages consisting of sets of stored procedures that interact with Web browsers through HTTP. Stored procedures can be executed in several ways:

- From a hypertext link that calls a stored procedure when it is selected
- By clicking **Submit** on an HTML form to denote the completion of a task such as filling out a form supplied on the HTML page
- By passing parameters to a stored procedure based on user choices from a list

Information in the stored procedure, such as tagged HTML text, is displayed in the Web browser as a Web page. These dynamic Web pages are generated by the database and are based on the database contents and the input parameters passed in to the stored procedure. Using PL/SQL stored procedures is especially efficient and powerful for generating dynamic Web page content.

Use Oracle Multimedia when media data such as images, audio, video, or combinations of all three are to be uploaded into and retrieved from database tables,

using the functions and procedures in the Oracle Multimedia PL/SQL packages or the Oracle Multimedia object types and their respective sets of methods.

Media upload procedures first perform a SQL INSERT operation to insert a row of data in the media table, which also initializes instances of the respective multimedia columns with an empty BLOB. Next, a SQL SELECT FOR UPDATE operation selects the multimedia columns for update. Finally a SQL UPDATE operation updates the multimedia columns. Oracle Multimedia procedures or methods are called to perform these tasks:

- Initialize the multimedia columns with an empty BLOB.
- Set attributes to indicate media data is stored internally in a BLOB, when using Oracle Multimedia object types.
- Get values of the multimedia attributes and store them.
- When exceptions occur, determine the length of the BLOB content and its MIME type.

Media retrieval operations involve these tasks:

- Retrieving the multimedia data from the database
- Checking the cache validity of the multimedia data based on its updated time in contrast to that of the HTTP header time
- Determining where the media data is located: in the database, in a BFILE, or at a URL location; then, getting the media, and downloading it for display on an HTML page

Related Topics

- [Oracle Multimedia PL/SQL Photo Album Sample Application](#)
The Oracle Multimedia PL/SQL Web Toolkit Photo Album sample application is a media upload and retrieval Web application using Oracle Multimedia object types. This application uses the PL/SQL Gateway and PL/SQL Web Toolkit.

See Also:

- *Oracle Application Express App Builder User's Guide* for information about using Oracle Application Express
- *Oracle Database Development Guide* for more information about using PL/SQL Gateway and PL/SQL Web Toolkit

2.5.1 Using the PL/SQL Gateway and PL/SQL Web Toolkit

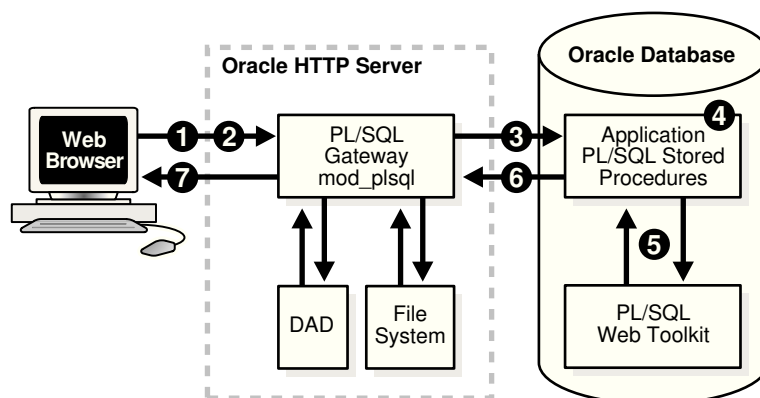
You can use the `mod_plsql` Gateway (a plug-in of Oracle HTTP Server) or the embedded version of the PLSQL Gateway that runs in the XML DB HTTP Listener in the database, to listen for browser requests, to execute stored PL/SQL procedures in the database, and to generate an HTML page containing data and code for the response returned to the Web browser for display.

Oracle HTTP Server serves mainly the static HTML files, images, and so on, that a Web application uses, and is usually located in the file system where Oracle HTTP Server is installed. Oracle HTTP Server contains modules or plug-ins that extend its

functions. One of these modules supplied by Oracle is the `mod_plsql` module, also known as the PL/SQL Gateway. The PL/SQL Gateway serves data dynamically from the database to Web browsers by calling PL/SQL stored procedures. The PL/SQL Gateway receives requests from a Web browser that are mapped to PL/SQL stored procedure calls. PL/SQL stored procedures retrieve data from the database and generate an HTTP response containing the data and code from the PL/SQL Web Toolkit to display the generated Web page in a Web browser. The PL/SQL Web Toolkit contains a set of packages (including `http`, `htf`, and `owa`) that can be used in the stored procedures to get information about the request, construct HTML tags, and return header information to the client Web browser.

The following figure shows these main components: Oracle HTTP Server, the Web browser, and the database.

Figure 2-1 Components of a PL/SQL Web Application



The numbers in the preceding figure represent steps that describe how a client Web browser request is turned into a Web page response from the execution of the PL/SQL procedure:

1. A client Web browser sends a PL/SQL server page or servlet request to Oracle HTTP Server.
2. Oracle HTTP Server routes the request to the PL/SQL Gateway (`mod_plsql`).
3. The PL/SQL Gateway forwards the request to the database using configuration information stored in the database access descriptor (DAD) and connects to the database.
4. The PL/SQL Gateway prepares the call parameters and invokes the PL/SQL package and the PL/SQL stored procedure in the application.
5. The PL/SQL procedure generates an HTML page using data from the database and special packages in the PL/SQL Web Toolkit accessed from the database. The PL/SQL Web Toolkit contains a set of packages, including `http`, `htf`, and `owa`, which are used in the stored procedures to get information about the request, construct HTML tags, and return header information to the client Web browser as the response returned to the PL/SQL Gateway.
6. The PL/SQL Gateway sends the response to Oracle HTTP Server.
7. Oracle HTTP Server sends the response to the client Web browser for display as a formatted Web page.

Example 2-4 URL Format to Invoke mod_plsql in a Web Browser

```
protocol://hostname[:port number]/DAD-name/[!][schema name.]  
[package name.]procedure_name[?query_string]
```

Example 2-5 URL Format to Invoke mod_plsql for the Photo Album Application

```
protocol://<hostname>[:<port-number>]/DAD-name/]procedure_name
```

Usually, the returned formatted Web page has one or more additional links, and each link, when selected, sends another request to the database through the PL/SQL Gateway to execute one or more stored procedures. The generated response displays data on the client Web page usually with additional links, which, when selected, execute more stored procedures that return the generated response for display as yet another formatted Web page, and so on.

Web application developers who use the PL/SQL Gateway create a PL/SQL package specification and body that describe procedures and functions that comprise the application. The package specification defines the procedures and functions used by the application, and the package body is the implementation of each procedure and function. All packages are compiled and stored in the database to perform specific operations for accessing data in the database and formatting HTML output for Web page presentation.

Oracle HTTP Server maps a URL entered in a browser to a specific PL/SQL procedure stored in the database. It does this by storing specific configuration information in a DAD for each stored procedure. Thus, each DAD contains the database connection information that the Web server requires to translate the URL entered into a database connection to call the stored procedure.

Oracle HTTP Server listens for a request, routes the request to the PL/SQL Gateway, which forwards it to the database. Configuration information values stored in a DAD determine the database alias to use, the connection string to use for remote access, the procedure to use for uploading or downloading documents, and the user name and password information to enable access to the database. From the Web browser, the user specifies the URL that invokes the PL/SQL Gateway. The URL has a defined format for specifying all the required and optional parameters, including the location of the DAD and the name of the PL/SQL stored procedure to run, as shown in [Example 2-4](#).

To use the Oracle Multimedia Photo Album sample application and the PL/SQL Web Toolkit, the URL can be simplified to the format shown in [Example 2-5](#).

When the URL is entered in the Web browser, it includes the protocol (HTTP or HTTPS), the name of the hosting Web server, and the port number on which it is listening to handle requests. Next, the specified virtual path includes `/pls/<DAD-name>` to indicate that the Web server is configured to invoke `mod_plsql`, and the location of the DAD on the Web server.

In [Example 2-4](#), the last five parameters include the exclamation point (!) character, schema name, package name, procedure name, and query string. From the syntax, the exclamation point, schema name, package name, and query string parameters are optional; only the procedure name is required.

The exclamation point indicates that flexible parameter passing is being used. The schema name, if omitted, is resolved based on the user name. The package name, if omitted, means the procedure is standalone. The query string parameters are for the stored procedure and follow a special format. Of these five parameters, the procedure

name must be specified in both the DAD and the URL. The other four parameters are specified in either the DAD or the URL, or not at all, depending on the application.

The URL displays the home page for the specified DAD. When the URL is entered in the address field of the Web browser page, it invokes either the specified DAD location only, or the specified DAD location along with the procedure name, or the specified DAD location along with the `schema.package.procedure` name. The response is returned as an HTML page. The HTML page contains the requested data and any other specified code for display in the client's Web browser. The Code Wizard demonstrates how this operation works. For example, to invoke the Code Wizard administration URL, enter the following URL:

```
http://<hostname>:<port-number>/pls/ordcwadmin
```

The virtual path includes `pls` to indicate that the Web server is configured to invoke `mod_plsql`, followed by the name of the DAD used for the Code Wizard administrator, `ordcwadmin`.

When the HTML page is displayed, it resolves to the following URL for the Code Wizard administrator:

```
http://<hostname>:<port-number>/pls/ordcwadmin/ORDCWPKG.menu
```

`ORDCWPKG.menu` represents the `package.procedure` name, which is specified as the default home page in the `ordcwadmin` DAD.

When the PL/SQL Gateway is invoked, it uses the stateless model and does not permit a transaction to span across multiple HTTP requests. In this stateless model, applications typically can create a session to maintain state by using one of these techniques: HTTP cookies, a hidden HTML field as an HTML form element of the HTML Form package, or storage of vital information in database tables for query.

Related Topics

- [Oracle Multimedia PL/SQL Photo Album Sample Application](#)
The Oracle Multimedia PL/SQL Web Toolkit Photo Album sample application is a media upload and retrieval Web application using Oracle Multimedia object types. This application uses the PL/SQL Gateway and PL/SQL Web Toolkit.
- [Oracle Multimedia Code Wizard Sample Application for the PL/SQL Gateway](#)
The Oracle Multimedia Code Wizard sample application for the PL/SQL Gateway creates PL/SQL procedures for the PL/SQL Gateway to upload and retrieve multimedia data stored in a database using Oracle Multimedia object types.

3

Oracle Multimedia PL/SQL Photo Album Sample Application

The Oracle Multimedia PL/SQL Web Toolkit Photo Album sample application is a media upload and retrieval Web application using Oracle Multimedia object types. This application uses the PL/SQL Gateway and PL/SQL Web Toolkit.

This application assumes the following:

- You are familiar with developing PL/SQL applications using the PL/SQL Gateway and PL/SQL Web Toolkit.
- You have installed and configured the Oracle Multimedia PL/SQL Web Toolkit Photo Album sample application.

You can install the Oracle Multimedia PL/SQL Web Toolkit Photo Album sample application from the Oracle Database Examples media, which is available for download from the Oracle Technology Network (OTN). After installing the Oracle Database Examples media, the sample application files and README.txt file are located at:

```
<ORACLE_HOME>/ord/http/demo/plsqlwtk (on Linux and UNIX)
```

```
<ORACLE_HOME>\ord\http\demo\plsqlwtk (on Windows)
```

This chapter describes how to run the PL/SQL Web Toolkit Photo Album sample application. See the `README.txt` file for this sample application for additional requirements and instructions on installing and configuring this sample application.

This chapter includes these sections:

- [Overview of the PL/SQL Photo Album Sample Application](#)
- [Running the PL/SQL Photo Album Sample Application](#)
- [Description of the PL/SQL Photo Album Sample Application](#)

See Also:

[Oracle Multimedia Code Wizard Sample Application for the PL/SQL Gateway](#) for a sample application that creates media upload and retrieval procedures for the PL/SQL Gateway

3.1 Overview of the PL/SQL Photo Album Sample Application

This topic describes the setup process for the Oracle Multimedia PL/SQL Web Toolkit Photo Album sample application.

The Oracle Multimedia PL/SQL Web Toolkit Photo Album sample application demonstrates how to perform the following operations:

- Use the Oracle Multimedia image object type to upload, retrieve, and process media data stored in Oracle Database.
- Combine the image metadata methods of Oracle Multimedia with the XML document management capabilities of Oracle XML DB and the full-text indexing and search features of Oracle Text to create a solution that can extract, store, and search metadata that is embedded in binary image files.
- Collect new metadata from a user, format the metadata into an XML document, and store the document in the binary image using the Oracle Multimedia image object type.

When installed, this photo album application creates several schema objects that are important to the following discussion. These objects include the `photos` table, which is defined by the following CREATE TABLE statement:

```
CREATE TABLE photos( id          NUMBER PRIMARY KEY,
                    description  VARCHAR2(40) NOT NULL,
                    metaORDImage XMLTYPE,
                    metaEXIF     XMLTYPE,
                    metaIPTC     XMLTYPE,
                    metaXMP      XMLTYPE,
                    image        ORDSYS.ORDIMAGE,
                    thumb        ORDSYS.ORDIMAGE )

--
-- store full-size and thumbnail images as SecureFiles LOBS
--
LOB(image.source.localdata) STORE AS SECUREFILE
LOB(thumb.source.localdata) STORE AS SECUREFILE
--
-- and bind the XMLType columns to the Oracle Multimedia metadata schemas
XMLType COLUMN metaORDImage
  STORE AS SecureFile CLOB
  XMLSCHEMA "http://xmlns.oracle.com/ord/meta/ordimage"
  ELEMENT "ordImageAttributes"
XMLType COLUMN metaEXIF
  STORE AS SecureFile CLOB
  XMLSCHEMA "http://xmlns.oracle.com/ord/meta/exif"
  ELEMENT "exifMetadata"
XMLType COLUMN metaIPTC
  STORE AS SecureFile CLOB
  XMLSCHEMA "http://xmlns.oracle.com/ord/meta/iptc"
  ELEMENT "iptcMetadata"
XMLType COLUMN metaXMP
  STORE AS SecureFile CLOB
  XMLSCHEMA "http://xmlns.oracle.com/ord/meta/xmp"
  ELEMENT "xmpMetadata";
```

The data types for the `image` and `thumb` columns are defined as Oracle Multimedia image object types. These columns are used to store the full-size images and the generated thumbnail images, respectively. The LOB storage clauses direct the database to store the full-size and thumbnail images in SecureFiles LOBs, which are the highest performing storage option for binary data.

The table also defines four columns of type `XMLType` to store XML documents that contain four different kinds of image metadata. Each column is bound to a specific Oracle Multimedia metadata schema. Each metadata schema defines precisely the data model of the metadata document. These schemas are registered with Oracle XML DB when the database is created. The column definitions specify that the database uses unstructured storage to manage the XML metadata documents. Some advantages of using unstructured storage to manage XML include fast retrieval of the complete document and the ability to use `XMLIndex` indexes to improve the performance of XPath-based queries.

When installed, this photo album application also creates other schema objects. These schema objects include two types of indexes that accelerate metadata searches: a `CONTEXT` text index and an `XMLIndex` index.

The `CONTEXT` type is a text index over all columns that contain descriptive information about the image. These columns include `PHOTOS.DESCRPTION`, which is a `VARCHAR2` data type, and these four `XMLType` columns: `PHOTOS.METAIPTC`, `PHOTOS.METAEXIF`, `PHOTOS.METAXMP`, and `PHOTOS.METAORDIMAGE`. The `CONTEXT` text index is used to accelerate metadata searches by implementing the photo album search feature that enables users to search for photographs by keyword or phrase.

The `CONTEXT` text index is created by the following statements. (This example assumes that this photo album application has been installed in the `SCOTT` schema.)

```
-- Create preference PA_CTXIDX.
ctx_ddl.create_preference('SCOTT.PA_CTXIDX', 'MULTI_COLUMN_DATASTORE');

-- Create a multicolumn datastore.
ctxcols := 'description, ' ||
           'SCOTT.photo_album.getClob(METAIPTC), ' ||
           'SCOTT.photo_album.getClob(METAEXIF), ' ||
           'SCOTT.photo_album.getClob(METAXMP), ' ||
           'SCOTT.photo_album.getClob(METAORDIMAGE)';
ctx_ddl.set_attribute( ctxpref, 'COLUMNS', ctxcols );

-- Create the CONTEXT text index.
create index pa_ctx_idx on photos(description)
indextype is ctxsys.context
parameters ( 'DATASTORE SCOTT.PA_CTXIDX' );
```

The `XMLIndex` index is used to accelerate metadata searches by permitting users to search only certain types of image metadata and limiting the search to specific portions of an XML document. For example, the following statements create three indexes of type `XMLIndex` to speed up `existsNode()` queries on columns of type `XMLType`:

```
create index pa_path_iptc_idx on photos( metaIptc )
indextype is XDB.XMLIndex;

create index pa_path_exif_idx on photos( metaExif )
indextype is XDB.XMLIndex;
```

```
create index pa_path_xmp_idx on photos( metaXMP )
  indextype is XDB.XMLIndex;
```

During the installation, as prescribed by the PL/SQL Gateway, a document upload table is defined by the following CREATE TABLE statement:

```
CREATE TABLE PHOTOS_UPLOAD( name          VARCHAR2(256) UNIQUE NOT NULL,
                             mime_type     VARCHAR2(128),
                             doc_size      NUMBER,
                             dad_charset   VARCHAR2(128),
                             last_updated  DATE,
                             content_type  VARCHAR2(128),
                             blob_content  BLOB )
--
-- store BLOBs as SecureFiles LOBs
--
LOB(blob_content) STORE AS SECUREFILE;
```

Each image uploaded using the PL/SQL Gateway is stored in the `PHOTOS_UPLOAD` table. An upload procedure (`insert_new_photo`) automatically moves the uploaded image from the specified `PHOTOS_UPLOAD` table to the photo album applications table called `photos`.

See Also:

- *Oracle XML DB Developer's Guide* for more information about XML DB and XMLIndex indexes
- *Oracle Text Application Developer's Guide* for more information about creating and using text indexing

3.2 Running the PL/SQL Photo Album Sample Application

This topic describes how to run the PL/SQL Photo Album application.

After you have completed the setup tasks and have built the PL/SQL Photo Album application, including creating a database access descriptor (DAD) entry (as described in the `README.txt` file), you are ready to run this application.

In the address field of your Web browser, enter the following URL:

```
<protocol><hostname:port-number>/photoalbum
```

1. In the `<protocol>` field, enter `http://`.
2. In the `<hostname:port-number>` field, enter the host name and port number of the system where your HTTP server is running.

When first invoked, this photo album application displays any images that are currently stored in the album. By default, the photo album is empty when first installed. To upload a new photograph, select **Upload photo**. Enter a description of the photograph and the name of the image file, or browse to its directory location. Then, click **Upload photo**.

The contents of the photo album are displayed, along with a picture of the new photograph. Click the thumbnail image to view the full-size version of the photograph. When this photo album application displays the text **view image** instead of its

thumbnail image, the image format that was uploaded was not recognized by Oracle Multimedia. Click **view image** to display the full-size image.

You can now begin to load your photo album application with your favorite photographs.

3.3 Description of the PL/SQL Photo Album Sample Application

The PL/SQL Photo Album application is implemented as a set of PL/SQL procedures and functions, organized in a single PL/SQL package.

These PL/SQL procedures and functions combine several database features to create the application. Oracle Multimedia is used to store and process image data. It is also used to extract metadata from images and embed new metadata into images. The XMLType feature is used to store and process the XML metadata documents. Oracle Text indexes are used to accelerate two kinds of metadata searches. Finally, the PL/SQL Web Toolkit is used to create HTML pages and deliver media content.

The user interface for the PL/SQL Photo Album application consists of a set of Web pages. You can use these Web pages to perform a set of tasks. The tasks and the Web pages are introduced in this topic and described in further detail in the following sections.

You can explore this photo album application using the navigation bar near the top of each Web task page. The leftmost entry of the navigation bar displays the name of the current Web page. On the right, there are links to other Web pages you can access from the current page. Each Web task page contains a link to the **View album** page, which is the home page for the application.

Pages in the PL/SQL Photo Album Sample Application

The following topics, which are summarized here, describe each page in the PL/SQL Photo Album application:

- [Browsing the Photo Album](#)

Use the **View album** page to display thumbnail-size versions of all the images in the photo album and a description link positioned under each thumbnail image. When you select a thumbnail image, the full-size image is displayed. When you select the description link for an image, all the metadata for that image is displayed. The **View album** page is the home page for the application.
- [Adding Images to the Photo Album](#)

Use the **Upload photo** page to display a simple form to collect a description for a new image, and the directory path to the location of the image on the local computer. When you click the **Upload photo** button, the browser sends the image to the Web server and the image is stored in the database.
- [Searching for Images by Keyword or Phrase](#)

Use the **Search album** page to display a search album form to collect keywords or phrases to initiate full-text searches through all image metadata. The application queries the database for all images with metadata that contains the specified keywords or phrases. The search results are displayed as a set of thumbnail images. The search album form is also available from the **View album** page.

- [Viewing Full-Size Images](#)

Use the **View entry** page to display the full-size image of a specified photograph, including any description text that was entered for that image when it was uploaded.
- [Examining Image Metadata](#)

Use the **View metadata** page to display all the metadata that was extracted from the image when it was uploaded. Up to four types of metadata can be displayed.
- [Writing New XMP Metadata to Images](#)

Use the **Write XMP metadata** page to display a form to collect input for five metadata attributes. These attributes are formatted into an XML document that is embedded within the binary image. The new XMP metadata overwrites any existing XMP metadata.
- [Searching for Images That Contain Specific Metadata Attributes](#)

Use the **Search metadata** page to collect input and perform advanced metadata searches. You can specify the type of metadata to be searched. Optionally, you can also limit the search to a specific XML tag within the specified document. The search results are displayed as a set of thumbnail images.

 **See Also:**

- *Oracle XML DB Developer's Guide* for more information about querying and updating XML metadata
- *Oracle Text Application Developer's Guide* for more information about text indexing
- *Oracle Database Advanced Application Developer's Guide* for more information about developing PL/SQL Web applications

3.3.1 Browsing the Photo Album

You can use the **View album** page of the photo album application to browse the contents of a photo album and to access other photo album capabilities.

The home page for this photo album application, **View album**, displays the contents of the photo album as thumbnail images in four-column format. Each thumbnail image is also a link to the **View entry** page. When you click a thumbnail image link, the application displays the full-size image on a View entry page. Included under each thumbnail image on the **View album** page is the image description that was entered when the image was uploaded to the album. The description is also a link to the **View metadata** page where all the metadata for this photograph can be examined.

Near the top of the **View album** page, there is a text entry field (in the shape of a rectangular box) that accepts user input for a full-text search through all the photo album metadata. The **Search** button to the right of the text field initiates the search. The search results are displayed on the **Search album** page.

At the top of the **View album** page, there is a navigation bar, which includes links to other photo album pages. From the **View album** page, you can navigate to the **Search metadata** page or the **Upload photo** page.

Figure 3-1 shows the **View album** page for an album that contains several images.

Figure 3-1 View album Page with Uploaded Images



The PL/SQL procedures `view_album`, `print_album`, `print_image_link`, and `deliver_media` are the primary application components that implement the **View album** page. The `view_album` procedure is a public procedure that takes a single optional argument. By default, the argument has a `NULL` value. Or, it can have the value of the string entered in the text entry field on the **Search album** page. When the search argument is `NULL`, the `SELECT` statement retrieves the `id`, `description`, and `thumb` columns for all entries in the `photos` table. When the search string is not `NULL`, the `SELECT` statement uses the `CONTAINS` operator to restrict the result set to only images with metadata that matches the search string.

Example 3-1 contains some relevant lines of code in the `view_album` procedure.

The `SELECT` statement is bound to the cursor variable `album_cur` and passed to the procedure `print_album`, which creates the HTML output.

The `print_album` procedure uses the HTP and HTF packages from the PL/SQL Web Toolkit to create the HTML tags that format the output into a four-column table. Each cell in the table contains two links or anchor tags. The first link is to the **View entry** page, which displays the full-size version of the image. This anchor is implemented by `PHOTO_ALBUM.VIEW_ENTRY`, and passes `entry_id` as a query string input argument. If the thumbnail image has a nonzero length, then procedure `print_image_link` is called to create an HTML `` tag that is the content (the thumbnail image) of the anchor link. The string `thumb` and the `entry_id` are passed to procedure `print_image_link`, along with the image description, and the height and width of the thumbnail image. These values are used to create the `` tag.

If an image is in a format that Oracle Multimedia does not support, the application cannot create a thumbnail version of the image. In this case, the content of the anchor link is the text **view image**.

Example 3-2 contains some relevant lines of code in the `print_album` procedure.

The procedure `print_image_link` uses the `height` and `width` arguments to populate the `height` and `width` attributes of the `` tag. The `description` argument is used to create text for the `alt` attribute. If the `description` argument is empty, a default string is constructed. Finally, the `src` attribute is set to the URL

PHOTO_ALBUM.DELIVER_MEDIA with two query string arguments, `media` and `entry_id`. The `media` argument controls whether the thumbnail or full-size version of the image is delivered. The `entry_id` argument identifies the image to be delivered.

Example 3-3 contains some relevant lines of code in the `print_image_link` procedure.

The procedure `deliver_media` fetches the image content from the database. The `If-Modified-Since` HTTP request header is compared to the last modification time of the image. If the image has not been modified, a response is sent that the browser can display the image from its cache. Otherwise, the image MIME type and last modified time are sent to the Web server, along with the image content.

Example 3-4 contains some relevant lines of code in the `deliver_media` procedure.

Example 3-1 Procedure view_album

```
--
-- no search criteria so fetch all entries
--
IF search IS NULL THEN
  OPEN album_cur FOR
    SELECT id, description, thumb
    FROM photos
    ORDER BY id;
  print_album( album_cur, 'The photo album is empty.' );
  CLOSE album_cur;
ELSE
--   -- use the full-text index to select entries matching the search criteria
--
  OPEN album_cur FOR
    SELECT id, description, thumb
    FROM photos
    WHERE CONTAINS( description, trim(search) ) > 0
    ORDER BY id;
  print_album( album_cur, 'No photos were found.' );
  CLOSE album_cur;
END IF;
```

Example 3-2 Procedure print_album

```
-- escape the description text
sc_description := htf.escape_sc( entry.description );

--
-- Display the thumbnail image as an anchor tag which can be used
-- to display the full-size image. If the image format is not
-- supported by Oracle Multimedia, then a thumbnail would not have been
-- produced when the image was uploaded, so use the text '[view
-- image]' instead of the thumbnail.
--

http.print( '<td headers="c' || colIdx || '" align="center" >
           <a href="PHOTO_ALBUM.VIEW_ENTRY?entry_id=' ||
           entry.id || '>' );
IF entry.thumb.contentLength > 0
THEN
  print_image_link( 'thumb', entry.id, sc_description,
                  entry.thumb.height, entry.thumb.width );
ELSE
  http.prn( '[view image]' );
END IF;
```

```

    http.print( '</a>' );

    -- Create link to the metadata
    http.prn('<br>');
    http.anchor( curl=>'PHOTO_ALBUM.VIEW_METADATA?entry_id=' || entry.id,
                ctext=>sc_description );
    http.prn('</td>');

```

Example 3-3 Procedure print_image_link

```

-- add height and width to tag if non zero
IF height > 0 AND width > 0 THEN
    attributes := attributes || ' height=' || height || ' width=' || width;
END IF;

-- create an alt text if none given
IF alt IS NULL THEN
    IF type = 'thumb' THEN
        alt2 := 'thumb-nail image ';
    ELSE
        alt2 := 'full-size image ';
    END IF;
    alt2 := alt2 || 'for album entry ' || entry_id;
ELSE
    alt2 := alt;
END IF;

http.img( curl=>'PHOTO_ALBUM.DELIVER_MEDIA?media=' || type ||
          ampersand || 'entry_id=' || entry_id,
          calt=>alt2, cattributes=>attributes );

```

Example 3-4 Procedure deliver_media

```

--
-- Fetch the thumbnail or full-size image from the database.
--
IF media = 'thumb'
THEN
    SELECT thumb INTO local_image FROM photos WHERE id = entry_id;
ELSE
    SELECT image INTO local_image FROM photos WHERE id = entry_id;
END IF;

--
-- Check update time if browser sent If-Modified-Since header
--
IF ordplsgwyutil.cache_is_valid( local_image.getUpdateTime() )
THEN
    owa_util.status_line( ordplsgwyutil.http_status_not_modified );
    RETURN;
END IF;

--
-- Set the MIME type and deliver the image to the browser.
--
owa_util.mime_header( local_image.mimeType, FALSE );
ordplsgwyutil.set_last_modified( local_image.getUpdateTime() );
owa_util.http_header_close();

IF owa_util.get_cgi_env( 'REQUEST_METHOD' ) <> 'HEAD' THEN

```

```

wpg_docload.download_file( local_image.source.localData );
END IF;

```

Related Topics

- [Overview of the PL/SQL Photo Album Sample Application](#)
This topic describes the setup process for the Oracle Multimedia PL/SQL Web Toolkit Photo Album sample application.

3.3.2 Adding Images to the Photo Album

You can use the **Upload photo** page of the photo album application to add more content to a photo album.

The **Upload photo** page is used to add new images to the photo album. The page displays a form with two text entry fields. In the **Description:** field, you can optionally enter a word or short phrase that describes the image. In the **File name:** field, enter the name of the image file or click **Browse...** to locate the image file to be uploaded. The **Upload photo** button under the **File name:** field starts the upload operation. When the image is successfully uploaded, the **View album** page appears. From that page, you can display the contents of the photo album.

At the top of the **Upload photo** page, there is a navigation bar, which includes links to other photo album pages. From the **Upload photo** page, you can return to the **View album** page or select the **Search metadata** page.

[Figure 3-2](#) shows an **Upload photo** page with all the entry fields completed.

Figure 3-2 Completed Upload photo Page

The PL/SQL procedures `view_upload_form`, `print_upload_form`, and `insert_new_photo` are the primary application components that implement the **Upload photo** page. Together, `view_upload_form` and `print_upload_form` create the HTML page that is displayed. The page contains a form tag, a portion of which is shown in [Example 3-5](#). The target of the form is `PHOTO_ALBUM.INSERT_NEW_PHOTO`.

[Example 3-5](#) contains some relevant lines of code in the `print_upload_form` procedure.

Procedure `insert_new_photo` receives the form, processes the inputs, and stores the new image in the database.

First, the `insert_new_photo` procedure checks that a file name was entered into the upload form. The image size, MIME type, and BLOB locator for the image content are selected from the document upload table, and the size is checked to ensure that the

image is not of zero length. If the `description` field is blank, a description is created using the file name.

Next, the `ORDSYS.ORDIMAGE()` constructor is called to initialize the `thumb` and `image` `ORDImage` object type columns with an empty BLOB for the new row to be stored in the `photos` table. A SQL `SELECT FOR UPDATE` statement fetches the newly initialized thumbnail image and full-size image object type columns for updating. A `DBMS_LOB.COPY` operation loads the image from the upload table into the `image` `ORDImage` object type column.

The `ORDImage` object method `setProperties()` reads the image and sets the image object attributes. Because some browsers cannot display some image formats inline, in this sample application, BMP formatted images are converted to a JPEG image format (for images with more than 8 bits of color), or a GIFF image format (for images with less than 9 bits of color) by calling the `get_preferred_format` function. A `processCopy()` operation is performed on the full-size image to create the thumbnail image.

The `ORDImage` object `getMetadata()` method is called to extract all supported types of image metadata. The root element of each XML document in the return vector is examined to discover the metadata type so that the documents can be stored in the correct columns.

Then, a SQL `UPDATE` statement stores the full-size image, the thumbnail image, and the image metadata documents in the database. Procedure `sync_indexes` is called to force an update of the text indexes. Finally, the form data input is deleted from the document upload table. A success message is returned to the browser, and the browser is redirected to the **View album** page.

[Example 3-6](#) contains some relevant lines of code in the `insert_new_photo` procedure.

Example 3-5 Procedure `print_upload_form`

```
<form action="PHOTO_ALBUM.INSERT_NEW_PHOTO"
method="post"
enctype="multipart/form-data">
database.
```

Example 3-6 Procedure `insert_new_photo`

```
--
-- Make sure a file name has been provided. If not, display an error
-- message, then re-display the form.
--
IF new_photo IS NULL OR LENGTH( new_photo ) = 0
THEN
    print_page_header;
    print_error( 'Please supply a file name.' );
    print_upload_form;
    print_page_trailer( TRUE );
    return;
END IF;

--
-- Get the length, MIME type and the BLOB of the new photo from the
-- upload table.
--
SELECT doc_size,
       mime_type,
       blob_content
```

```

INTO   upload_size,
        upload_mime_type,
        upload_blob
FROM   photos_upload
WHERE  name = new_photo;

--
-- Make sure we have a valid file.
--
IF upload_size = 0
THEN
    print_page_header;
    print_heading( 'Error message' );
    http.print( '<hr size="-1"><p>Please supply a valid image file.</p>' );
    print_upload_form;
    print_page_trailer( TRUE );
    return;
END IF;

--
-- If the description is blank, then use the file name.
--
IF c_description IS NULL
THEN
    c_description := new_photo;
    pos := INSTR( c_description, '/', -1 );
    IF pos > 0
    THEN
        c_description := SUBSTR( c_description, pos + 1 );
    END IF;
    c_description := SUBSTR( 'Image from file: ' ||
                            c_description || '.', 1, 40 );
END IF;

--
-- Insert a new row into the table, returning the newly allocated sequence
-- number.
INSERT INTO photos ( id, description, metaExif, metaIPTC, metaXMP,
                    image, thumb )
VALUES ( photos_sequence.nextval, c_description, NULL, NULL, NULL,
        ORDSYS.ORDIMAGE(), ORDSYS.ORDIMAGE() )
RETURN id
INTO new_id;

--
-- Fetch the newly initialized full-size and thumbnail image objects.
--
SELECT image,
        thumb
INTO new_image,
        new_thumb
FROM photos
WHERE id = new_id
FOR UPDATE;

--
-- Load the photo from the upload table into the image object.
--
DBMS_LOB.COPY( new_image.source.localData, upload_blob, upload_size );
new_image.setLocal();

--
-- Set the properties. If the image format is not recognized, then

```

```

-- the exception handler will set the MIME type and length from the
-- upload table.
--
BEGIN
    new_image.setProperties();
EXCEPTION
    WHEN OTHERS THEN
        new_image.contentLength := upload_size;
        new_image.mimeType := upload_mime_type;
END;

--
-- Some image formats are supported by Oracle Multimedia but cannot be
-- displayed inline by a browser. The BMP format is one example.
-- Convert the image to a GIF or JPEG based on number of colors in the
-- image.
--
IF new_image.contentFormat IS NOT NULL AND
( new_image.mimeType = 'image/bmp' OR
  new_image.mimeType = 'image/x-bmp' )
THEN
    BEGIN
        new_image.process(
            'fileFormat=' ||
            get_preferred_format( new_image.contentFormat ) );
    EXCEPTION
        WHEN OTHERS THEN
            NULL;
    END;
END IF;

--
-- Try to copy the full-size image and process it to create the thumbnail.
-- This may not be possible if the image format is not recognized.
--
BEGIN
    new_image.processCopy( thumb_scale, new_thumb );
EXCEPTION
    WHEN OTHERS THEN
        new_thumb.deleteContent();
        new_thumb.contentLength := 0;
END;

--
-- fetch the metadata and sort the results
--
BEGIN
    metav := new_image.getMetadata( 'ALL' );
    FOR i IN 1..metav.count() LOOP
        meta_root := metav(i).getRootElement();
        CASE meta_root
            WHEN 'ordImageAttributes' THEN xmlORD := metav(i);
            WHEN 'xmpMetadata' THEN xmlXMP := metav(i);
            WHEN 'iptcMetadata' THEN xmlIPTC := metav(i);
            WHEN 'exifMetadata' THEN xmlEXIF := metav(i);
            ELSE NULL;
        END CASE;
    END LOOP;
EXCEPTION
    WHEN OTHERS THEN
        NULL;
END;

```

```
--
-- Update the full-size and thumbnail images in the database.
-- Update metadata columns
--
UPDATE photos
SET image = new_image,
    thumb = new_thumb,
    metaORDImage = xmlORD,
    metaEXIF = xmlEXIF,
    metaIPTC = xmlIPTC,
    metaXMP = xmlXMP
WHERE id = new_id;

-- -- update the text indexes
-- sync_indexes;

--
-- Delete the row from the upload table.
--
DELETE FROM photos_upload WHERE name = new_photo;
COMMIT;

--
-- Redirect browser to display full album.
-- print_page_header(
--   '<meta http-equiv="refresh" content="2;url=PHOTO_ALBUM.VIEW_ALBUM">' );
print_heading( 'Photo successfully uploaded into photo album' );
```

3.3.3 Searching for Images by Keyword or Phrase

You can use the **View album** and **Search album** pages of the photo album application to search for content in a photo album.

You can use the **View album** and **Search album** pages to perform a keyword or phrase search of the metadata stored in the photo album. On either of these pages, enter the keyword or phrase in the **Full text search:** text entry field and click **Search**. This photo album application uses the `CONTEXT` text index to locate images that have metadata containing the text you entered. If the search is successful, the thumbnail versions of the matching images are displayed in a four-column table. Select the thumbnail image to view the full-size version, or select the description link below the thumbnail image to view the metadata for the image. If the search fails, the message "No photos were found" is displayed.

At the top of the **Search album** page, there is a navigation bar, which includes links to other photo album pages. From the **Search album** page, you can return to the **View album** page or select the **Search metadata** or **Upload photo** pages.

[Figure 3-3](#) shows a **Search album** page that contains the results of a successful search operation.

Figure 3-3 Search album Page Showing Results



Full-text searching of the photo album is implemented by the `view_album` procedure.

Related Topics

- [Browsing the Photo Album](#)
You can use the **View album** page of the photo album application to browse the contents of a photo album and to access other photo album capabilities.

3.3.4 Viewing Full-Size Images

You can use the **View entry** page of the photo album application to display full-size images and their descriptions in a photo album.

When you select a thumbnail image, the application directs you to the **View entry** page. This page displays the description of the image and the full-size version of the image.

At the top of the **View entry** page, there is a navigation bar, which includes links to other photo album pages. From the **View entry** page, you can return to the **View album** page, or select any of the **View metadata**, **Write metadata**, **Search metadata**, or **Upload photo** pages.

[Figure 3-4](#) shows a **View entry** page that contains the description and the full-size version of an image.

Figure 3-4 View entry Page with a Full-Size Image



The PL/SQL procedures `view_entry`, `print_image_link`, and `deliver_media` are the primary application components that implement the **View entry** page. The procedure `view_entry` takes a single parameter, `entry_id`, which uniquely locates the image in the `photos` table. The description and image object are fetched from the `photos` table. The procedure `print_image_link` creates the HTML `` tag, and then calls procedure `deliver_media` to fetch the image content.

[Example 3-7](#) contains some relevant lines of code in the `view_entry` procedure.

Example 3-7 Procedure `view_entry`

```
--
-- Fetch the row.
--
BEGIN
  SELECT htf.escape_sc(description), image
  INTO sc_description, photo
  FROM photos
  WHERE id = entry_id;
  EXCEPTION
  WHEN no_data_found THEN
    print_error( 'Image <b>' || htf.escape_sc(entry_id) ||
                '</b> was not found.</p>' );
    print_page_trailer( TRUE );
    return;
END;

print_image_link( 'image', entry_id, sc_description,
                  photo.height, photo.width );
```

Related Topics

- [Browsing the Photo Album](#)
You can use the **View album** page of the photo album application to browse the contents of a photo album and to access other photo album capabilities.

3.3.5 Examining Image Metadata

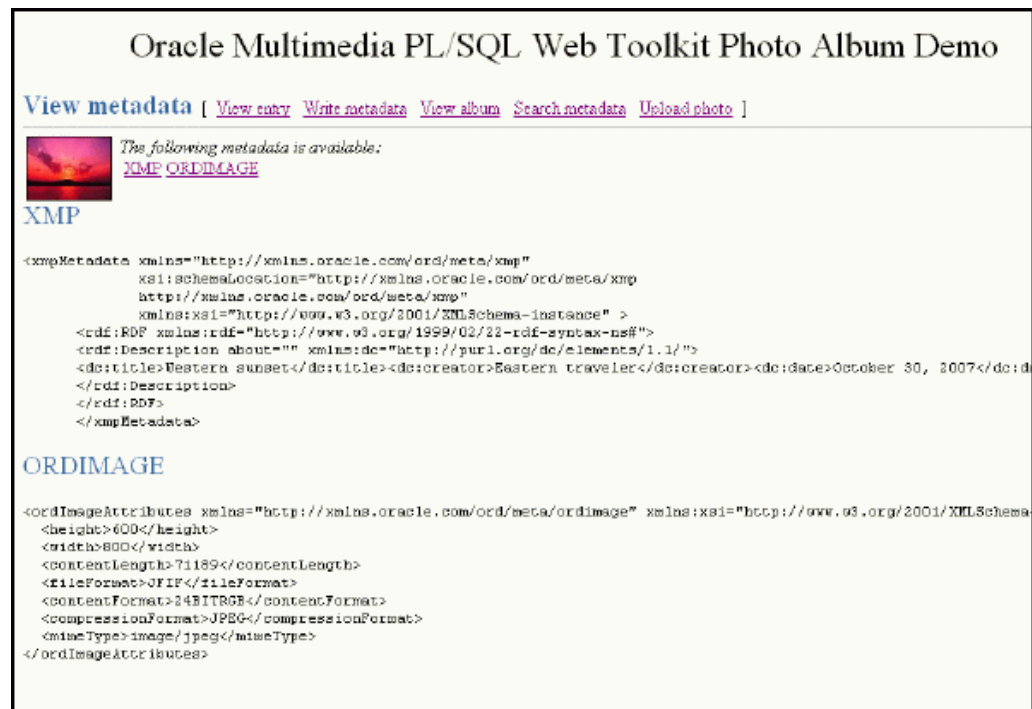
You can use the **View metadata** page of the photo album application to display the metadata for images in a photo album.

You can use the **View metadata** page to examine all the metadata for a specific image. Typically, you access this page from the **View album** page by selecting the description link below a thumbnail image. You can also access this page by selecting the **View metadata** link from the navigation bar. The **View metadata** page displays the thumbnail version of the image. To the right of the thumbnail image, there is a list of the metadata documents for this image. Each entry in the list is a link that takes you to the metadata document on the **View metadata** page.

At the top of the **View metadata** page, there is a navigation bar, which includes links to other photo album pages. From the **View metadata** page, you can return to the **View album** page, or select any of the **View entry**, **Write metadata**, **Search metadata**, or **Upload photo** pages.

Figure 3-5 shows a **View metadata** page that contains two types of metadata (XMP and ORDIMAGE) for an image.

Figure 3-5 View metadata Page with Metadata for an Uploaded Image



The PL/SQL procedures `view_metadata` and `print_metadata` are the primary application components that implement the **View metadata** page. The procedure `view_metadata` is

passed the argument `entry_id`, which uniquely identifies the image in the `photos` table. A `SELECT` statement retrieves all the `XMLType` metadata columns for the specified entry. If the metadata column is not `NULL`, procedure `print_metadata` is called to display the XML document inside an HTML `<pre>` tag.

[Example 3-8](#) contains some relevant lines of code in the `view_metadata` procedure.

The `print_metadata` procedure accepts an `XMLType` document as an argument. It uses the `getClobVal()` method to access the document as a `CLOB`. The content of the `CLOB` is read in a loop and formatted in the HTML page using the `http.prints` procedure. The `http.prints` procedure escapes the `'<'` and `'>'` characters so that they are rendered properly by the Web browser.

[Example 3-9](#) contains some relevant lines of code in the `print_metadata` procedure.

Example 3-8 Procedure `view_metadata`

```
--
-- Fetch the row.
--
SELECT metaOrdImage, metaEXIF, metaIPTC, metaXMP
INTO metaO, metaE, metaI, metaX
FROM photos
WHERE id = entry_id;

-- display the EXIF metadata
IF metaE IS NOT NULL THEN
  http.print( '<span class="bigBlue" id="exifMetadata">EXIF</span>' );
  http.print( '<br><pre>' );
  print_metadata( metaE );      http.print( '</pre>' );
END IF;
```

Example 3-9 Procedure `print_metadata`

```
metaClob := meta.getClobVal();
len := dbms_lob.getLength( metaClob );
IF bufSize > len THEN
  bufSize := len;
END IF;
WHILE len > 0 LOOP
  dbms_lob.read( metaClob, bufSize, pos, buf );
  http.prints( buf );
  pos := pos + bufSize;
  len := len - bufSize;
END LOOP;
```

3.3.6 Writing New XMP Metadata to Images

You can use the **Write XMP metadata** page of the photo album application to add or revise the XMP metadata for images in a photo album.

You can use the **Write XMP metadata** page to write new or replace existing XMP metadata in an image. Oracle Multimedia provides support for writing XMP metadata only. You can access the **Write XMP metadata** page by selecting the **Write metadata** link in the navigation bar from either the **View entry** page or the **View metadata** page.

The **Write XMP metadata** page displays the thumbnail version of the image to be modified. The page also displays an input form to collect metadata attributes in these five text entry fields:

- **Title:** Specify a title for the photograph.
- **Creator:** Enter the name of the person who took the photograph. This field is optional.
- **Date:** Enter the date the photograph was taken. This field is optional.
- **Description:** Enter a description, such as the subject of the photograph. This field is optional.
- **Copyright:** Enter the month and year when the photograph was taken. This field is optional.

Click **Write it!** to send the form to the application and embed the metadata in XMP format in the image.

At the top of the **Write XMP metadata** page, there is a navigation bar, which includes links to other photo album pages. From the **Write XMP metadata** page, you can return to the **View album** page, or select any of the **View entry**, **View metadata**, **Search metadata**, or **Upload photo** pages.

Figure 3-6 shows a **Write XMP metadata** page with completed entries for an image.

Figure 3-6 Completed Write XMP metadata Page with XMP Metadata for an Uploaded Image

The PL/SQL procedure `write_metadata` receives the form input fields from the browser. The procedure creates an XML document (as a string buffer) that is valid to the Oracle Multimedia XMP schema `http://xmlns.oracle.com/ord/meta/xmp`. The string buffer is used to create an XMLType object.

A `SELECT FOR UPDATE` statement retrieves the image to be modified. The Oracle Multimedia method `putMetadata()` is called to embed the XML document into the image. The modified image is stored back to the photos table. Finally, procedure `sync_indexes` is called to update the text indexes.

Example 3-10 contains some relevant lines of code in the `write_metadata` procedure.

The input data shown in **Example 3-10** would result in the storage of the following metadata in the image:

```
<xmpMetadata xmlns="http://xmlns.oracle.com/ord/meta/xmp"
             xsi:schemaLocation="http://xmlns.oracle.com/ord/meta/xmp
```

```

        http://xmlns.oracle.com/ord/meta/xmp"
        xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
<rdf:RDF xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#">
  <rdf:Description about="" xmlns:dc="http://purl.org/dc/elements/1.1/">
    <dc:title>Story time</dc:title>
    <dc:creator>father</dc:creator>
    <dc:date>July 4, 2001</dc:date>
    <dc:description>family reading</dc:description>
    <dc:copyright>mother</dc:copyright>
  </rdf:Description>
</rdf:RDF>
</xmpMetadata>

```

Example 3-10 Procedure write_metadata

```

-- Create the XMP packet it must be schema valid
-- to "http://xmlns.oracle.com/ord/meta/xmp"
-- and contain an <RDF> element. This example uses
-- the Dublin Core schema as implemented by Adobe XMP
buf := '<xmpMetadata xmlns="http://xmlns.oracle.com/ord/meta/xmp"
      xsi:schemaLocation="http://xmlns.oracle.com/ord/meta/xmp
      http://xmlns.oracle.com/ord/meta/xmp"
      xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" >
<rdf:RDF xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#">
<rdf:Description about="" xmlns:dc="http://purl.org/dc/elements/1.1/">
<dc:title>' || htf.escape_sc(title) || '</dc:title>';

IF c_creator IS NOT NULL THEN
  buf := buf || '<dc:creator>' || htf.escape_sc(c_creator)
          || '</dc:creator>';
END IF;
IF c_date IS NOT NULL THEN
  buf := buf || '<dc:date>' || htf.escape_sc(c_date)
          || '</dc:date>';
END IF;
IF c_description IS NOT NULL THEN
  buf := buf || '<dc:description>' || htf.escape_sc(c_description)
          || '</dc:description>';
END IF;
IF c_copyright IS NOT NULL THEN
  buf := buf || '<dc:copyright>' || htf.escape_sc(c_copyright)
          || '</dc:copyright>';
END IF;
buf := buf || '
</rdf:Description>
</rdf:RDF>
</xmpMetadata>';

xmp := XMLType.createXML(buf, 'http://xmlns.oracle.com/ord/meta/xmp');

-- -- select image for update
-- description is selected to force update of CTX index
--
SELECT image, description
INTO img, des
FROM photos
WHERE id = entry_id
FOR UPDATE;

--
-- write the metadata

```

```

--
img.putMetadata( xmp, 'XMP' );

--
-- save updated image and new metadata to table
-- description updated to force update of CTX index
--
UPDATE photos
SET image = img,
    metaXMP = xmp,
    description = des
WHERE id = entry_id;

-- update the text indexes
sync_indexes;

```

3.3.7 Searching for Images That Contain Specific Metadata Attributes

You can use the **Search metadata** page of the photo album application to search for specific metadata for images in a photo album.

You can use the **Search metadata** page to search a specific metadata type and to limit your search to a specific tag within a metadata document. You can access the **Search metadata** page by selecting the **Search metadata** link in the navigation bar of any photo album application Web page.

The **Search metadata** page displays a form with four fields to define how the search is to be performed. Use the menu in the **Search in metadata:** field to select the type of metadata (**EXIF**, **IPTC**, or **XMP**) to be searched. When this field is changed, the fields **Search in tag:** and **Search method:** are initialized with values that are appropriate to the type of metadata search.

Use the drop-down list in the **Search in tag:** field to limit the search to a specific XML element within a metadata document. The list is populated with element names that are appropriate for the selected metadata type. When the value **--Any tag--** is showing, the search looks at all elements within the document type. When the **XMP** metadata type is selected, searches are limited to Description elements within the parent RDF element. If the metadata document is properly constructed, selecting **RDF/Description** in this field searches all relevant metadata within XMP documents.

In the **Search method:** field, select **Contains** to specify a search where an element contains the search string. Select **Equals** to specify a search where element values are matched exactly to the search string. For searches in XMP metadata, only the **Contains** search method is available.

Finally, enter a keyword or phrase in the **Search string:** field and click **Search**. If the search is successful, the thumbnail versions of the matching images are displayed in a four-column table. Select the thumbnail image to view the full-size version of an image. Or, select the description link below the thumbnail image to view the metadata for the image. If the search fails, the message "No photos matched the search criteria." is displayed.

At the top of the **Search metadata** page, there is a navigation bar, which includes links to other photo album pages. From the **Search metadata** page, you can return to the **View album** page or select the **Upload photo** page.

Figure 3-7 shows a **Search metadata** page that contains sample search criteria and results from a successful search operation.

Figure 3-7 Completed Search metadata Page for an Uploaded Image

The PL/SQL procedure `search_metadata` receives the form input fields from the Web browser. The search parameters are used to build a query to find images that contain the desired metadata. The search is accomplished using the SQL function `XMLeXists`. The `XMLeXists` function is used to search an XML document for content that matches a given XQuery expression. The function returns `TRUE` if the document matched the search, and `FALSE` otherwise.

For example, assume that the `search_metadata` procedure receives input that specifies to search the `caption` tag in IPTC metadata for an exact match of the word "farm". The query to accomplish this search is as follows:

```
SELECT id, description, thumb
FROM photos
WHERE xmlexists('declare default element namespace ' ||
                ' "http://xmlns.oracle.com/ord/meta/iptc"; $x' ||
                '/iptcMetadata[//caption="farm"]' passing metaIptc as "x");
```

The XPath component of the XQuery expression, `/iptcMetadata[//caption="farm"]`, specifies a search for all `<caption>` elements under the root element `<iptcMetadata>` where the `<caption>` content is "farm".

See Also:

Oracle XML DB Developer's Guide for more information about the `XMLeXists` function

Example 3-11 contains some relevant lines of code in the `search_metadata` procedure.

Example 3-11 Procedure `search_metadata`

```
-- Set up search variables for EXIF documents.
IF mtype = 'exif' THEN
```



```

IF op = 'equals' THEN
  xpath := '/exifMetadata[/' || tag || '=' || c_search || '']';
ELSE -- default to contains
  xpath := '/exifMetadata/' || tag ||
    '[contains(., ' || c_search || ')]';
END IF;

xquery := 'declare default element namespace ' ||
  ' "http://xmlns.oracle.com/ord/meta/exif"; $x' || xpath;

OPEN album_cur FOR
  SELECT id, description, thumb
  FROM photos
  WHERE xmlexists(xquery passing metaExif as "x");

-- Set up search variables for IPTC documents.
ELSIF mtype = 'iptc' THEN
  IF op = 'equals' THEN
    xpath := '/iptcMetadata[/' || tag || '=' || c_search || '']';
  ELSE -- default to contains
    xpath := '/iptcMetadata/' || tag ||
      '[contains(., ' || c_search || ')]';
  END IF;

  xquery := 'declare default element namespace ' ||
    ' "http://xmlns.oracle.com/ord/meta/iptc"; $x' || xpath;

  OPEN album_cur FOR
    SELECT id, description, thumb
    FROM photos
    WHERE xmlexists(xquery passing metaIptc as "x");

-- Set up search variables for XMP documents.
ELSIF mtype = 'xmp' THEN
  -- default to contains
  xpath := '//rdf:Description//*[contains(., ' ||
    | c_search || ')]';

  -- Add rdf namespace prefix.
  xquery := 'declare namespace rdf = ' ||
    ' "http://www.w3.org/1999/02/22-rdf-syntax-ns#"; ' ||
    'declare default element namespace ' ||
    ' "http://xmlns.oracle.com/ord/meta/xmp"; $x' || xpath;

  OPEN album_cur FOR
    SELECT id, description, thumb
    FROM photos
    WHERE xmlexists(xquery passing metaXMP as "x");

ELSE
  errorMsg := 'Search domain is invalid: ' || htf.escape_sc(mtype);
END IF;

print_search_form( mtype, tag, op, c_search );
htp.print('<hr size="-1">');
print_album( album_cur, 'No photos matched the search criteria.' );

```

4

Oracle Multimedia Code Wizard Sample Application for the PL/SQL Gateway

The Oracle Multimedia Code Wizard sample application for the PL/SQL Gateway creates PL/SQL procedures for the PL/SQL Gateway to upload and retrieve multimedia data stored in a database using Oracle Multimedia object types.

This application assumes the following:

- You are familiar with developing PL/SQL applications using the PL/SQL Gateway.
- You have installed and configured the Oracle Multimedia Code Wizard sample application.

You can install the Oracle Multimedia Code Wizard sample application from the Oracle Database Examples media, which is available for download from the Oracle Technology Network (OTN). After installing the Oracle Database Examples media, the sample application files and `README.txt` file are located at:

```
<ORACLE_HOME>/ord/http/demo/plsgwycw (on Linux and UNIX)
```

```
<ORACLE_HOME>\ord\http\demo\plsgwycw (on Windows)
```

This chapter describes how to run the Code Wizard sample application. See the `README.txt` file for additional requirements and instructions on installing and configuring this sample application.

This chapter includes these sections:

- [Running the Code Wizard Sample Application](#)
- [Description of the Code Wizard Sample Application](#)
- [Sample Session: Using Images](#)
- [Known Restrictions of the Oracle Multimedia Code Wizard](#)

See Also:

[Oracle Multimedia PL/SQL Photo Album Sample Application](#) for a Photo Album sample Web application that uses PL/SQL scripts to upload and retrieve media using Oracle Multimedia object types

4.1 Running the Code Wizard Sample Application

This topic describes how to run the Code Wizard application.

To use the Code Wizard sample application to create and test media access procedures, you must perform these steps:

1. Create a new database access descriptor (DAD) or choose an existing DAD for use with the Code Wizard.
2. Authorize use of the DAD using the Code Wizard's administration function.
3. Create and test media upload and retrieval procedures.

4.2 Description of the Code Wizard Sample Application

You can use the Code Wizard sample application to create PL/SQL stored procedures for the PL/SQL Gateway to upload and retrieve media data (images, audio, video, and general media) stored in a database.

The Oracle Multimedia Code Wizard sample application uses these Oracle Multimedia object types and their respective methods:

- `ORDImage`
- `ORDAudio`
- `ORDVideo`
- `ORDDoc`

The Code Wizard guides you through a series of self-explanatory steps to create either a media retrieval procedure or a media upload procedure. You can create and compile standalone media access procedures. Or, you can create the source of media access procedures for inclusion in a PL/SQL package. Finally, after creating media access procedures, you can customize them to meet your specific application requirements.

These processes are similar to how the Oracle Multimedia PL/SQL Web Toolkit Photo Album application uses the `insert_new_photo` procedure as the image upload procedure, and the `deliver_media` procedure as the image retrieval procedure.

The following subsections describe how to use the Code Wizard application:

- [Creating a New DAD or Choosing an Existing DAD](#)
- [Authorizing a DAD](#)
- [Creating and Testing Media Upload and Retrieval Procedures](#)
- [Creating a Media Upload Procedure](#)
- [Creating a Media Retrieval Procedure](#)
- [Using the PL/SQL Gateway Document Table](#)
- [How Time Zone Information Is Used to Support Browser Caching](#)

Related Topics

- [Overview of the PL/SQL Photo Album Sample Application](#)
This topic describes the setup process for the Oracle Multimedia PL/SQL Web Toolkit Photo Album sample application.

4.2.1 Creating a New DAD or Choosing an Existing DAD

To create media upload or retrieval procedures, you must select one or more database access descriptors (DADs) for use with the Code Wizard. To prevent the unauthorized browsing of schema tables and to prevent the unauthorized creation of media access

procedures, you must authorize each DAD using the Code Wizard administration function.

Depending on your database and application security requirements, you can create and authorize one or more new DADs specifically for use with the Code Wizard. Or, you can authorize the use of one or more existing DADs.

Oracle recommends that any DAD authorized for use with the Code Wizard employ some form of user authentication mechanism. The simplest approach is to create or use a DAD that uses database authentication. To use this approach, select **Basic Authentication Mode** and omit the password in the DAD specification. Alternatively, you can use a DAD that specifies an existing application-specific authentication mechanism.

The following example describes how to create a DAD that enables you to create and test media upload and retrieval procedures in the `SCOTT` schema.

 **Note:**

To test media upload procedures, you must specify the name of a document table in the DAD. When testing an upload procedure, you can choose either the DAD you used to create the procedure or the DAD you used to access the application. You can choose a document table name when you create a DAD, edit a DAD to specify the document table name at a later time, or use an existing DAD that specifies a document table name. This example shows how to specify the document table name when you create the DAD.

1. Set your Web browser to the Oracle HTTP Server Home page. Select **PL/SQL Properties** in the Administration page to open the `mod_plsql` Services page.
2. Scroll to the DAD Status section on the `mod_plsql` Services page. Then, click **Create** to open the DAD Type page.
3. Select the DAD type to be **General**. Then, click **Next** to open the Database Connection page.
4. Enter `/scottcw` in the DAD Name field. Enter `SCOTT` for the database account, and leave the password blank. Enter the connection information in the Database Connectivity Information section. Enter `ORDCWPKG.MENU` in the Default page field, and leave the other fields blank. Then, click **Next** to open the Document, Alias, and Session page.
5. Enter `MEDIA_UPLOAD_TABLE` for the Document Table on the Document, Alias, and Session page. Then, click **Apply**.
6. Restart Oracle HTTP Server for the changes to take effect.

 **See Also:**

Oracle Database Development Guide for more information about configuring DADs

4.2.2 Authorizing a DAD

To use the Code Wizard, you must first authorize a database access descriptor (DAD).

To authorize a DAD for use with the Code Wizard, perform these steps:

1. Enter the Code Wizard's administration URL into the location bar for your browser. For example:

`http://<host-name>:<port-number>/ordcwadmin`

2. Enter the user name and password when prompted by the browser.
3. Select **DAD authorization** from the **Main menu**, as shown in the following figure. Then, click **Next**.

Figure 4-1 Main Menu for the Code Wizard

Oracle Multimedia Code Wizard for the PL/SQL Gateway

Main menu

Current DAD: ORDCWADMIN
Current schema: CWADMIN

Select the required function, then click the Next button.

- Create media retrieval procedure
- Create media upload procedure
- Change DAD
 - Change to ORDCWADMIN
 - Change to SCOTTCW
- DAD authorization
- Logout

Select action:

4. Enter the name of the DAD you want to authorize along with the user name, as shown in the following figure. Then, click **Apply**.

Figure 4-2 Authorize the SCOTTCW DAD

Oracle Multimedia Code Wizard for the PL/SQL Gateway

Authorize DADs for use with the Oracle Multimedia Code Wizard for the PL/SQL Gateway

The following table list the DADs and users currently authorized to use the Oracle Multimedia Code Wizard for the PL/SQL Gateway. To delete a DAD, check the corresponding checkbox. Note that the code wizard administration DAD, ORDCWADMIN, cannot be deleted.

DAD name	User name	Delete
ORDCWADMIN	CWADMIN	<input type="checkbox"/>

Enter a DAD name and user name below to authorize a DAD for use with the Oracle Multimedia Code Wizard for the PL/SQL Gateway.

DAD name:

User name:

Select action:

 **Note:**

Duplicate DADs are not permitted, and each authorized DAD must indicate which database schema the user is authorized to access with the Code Wizard, using the DAD. Use this same page to delete the authorization for any existing DADs that no longer require the Code Wizard.

5. Review the updated list of DADs that are authorized to use the Oracle Multimedia Code Wizard, as shown in the following figure. Then, click **Done**.

Figure 4-3 List of Authorized DADs



6. Select **Logout** from the **Main menu** to log out (clear HTTP authentication information), then click **Next**. The log out operation redirects the request to the PL/SQL Gateway built-in `logmeoff` function.



See Also:

Oracle Database Development Guide for more information about configuring DADs

4.2.3 Creating and Testing Media Upload and Retrieval Procedures

After you specify and authorize a database access descriptor (DAD), you are ready to run this application to create media upload and retrieval procedures.

To start the Code Wizard after completing all the setup tasks, follow these steps:

1. Enter the appropriate URL into the address field of your Web browser.

For example:

```
http://<hostname>:<port-number>/scottcw
```

or

```
http://<hostname>:<port-number>/mediadad/ordcwpkg.menu
```

2. Enter the user name and password when prompted by the browser. The **Main menu** page of the Oracle Multimedia Code Wizard for the PL/SQL Gateway is displayed, as shown in the following figure.

Figure 4-4 Use the SCOTTCW DAD

3. If the DAD is configured specifically for use with the Code Wizard, enter the DAD name. To use another DAD, enter the DAD name along with the Code Wizard package name and **Main menu** procedure name (`ORDCWPKG.MENU`) after the DAD name.
4. After logging in, you can log out (clear HTTP authentication information) at any time by selecting **Logout** from the **Main menu** page, then clicking **Next**. The logout operation redirects the request to the PL/SQL Gateway built-in `logmeoff` function.

To create a media upload procedure or a media retrieval procedure, select the appropriate option from the **Main menu** page, then click **Next**. The Code Wizard then guides you through a series of self-explanatory steps to create the procedure.

If you create a standalone media upload or retrieval procedure, you will have the opportunity to view the contents of the procedure and test it.

Related Topics

- [Sample Session: Using Images](#)
This sample session uses the `SCOTT` schema to demonstrate the creation of image media upload and retrieval procedures. To use a different schema, substitute a different schema name and password. Or, if you have changed the password for the `SCOTT` schema, use your new password.
- [Creating a Media Upload Procedure](#)
Follow this process to create a media upload procedure using the Oracle Multimedia Code Wizard for the PL/SQL Gateway.
- [Creating a Media Retrieval Procedure](#)
Follow this process to create a media retrieval procedure using the Oracle Multimedia Code Wizard for the PL/SQL Gateway.

 **See Also:**

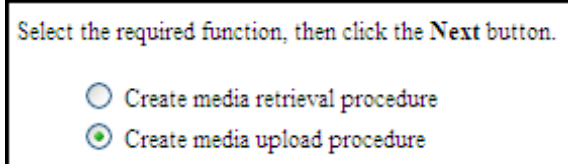
Oracle Database Development Guide for more information about using the embedded PL/SQL Gateway

4.2.4 Creating a Media Upload Procedure

Follow this process to create a media upload procedure using the Oracle Multimedia Code Wizard for the PL/SQL Gateway.

1. Select **Create media upload procedure** from the **Main menu** page, as shown in [Figure 4-5](#). Then, click **Next**.

Figure 4-5 Create a Media Upload Procedure

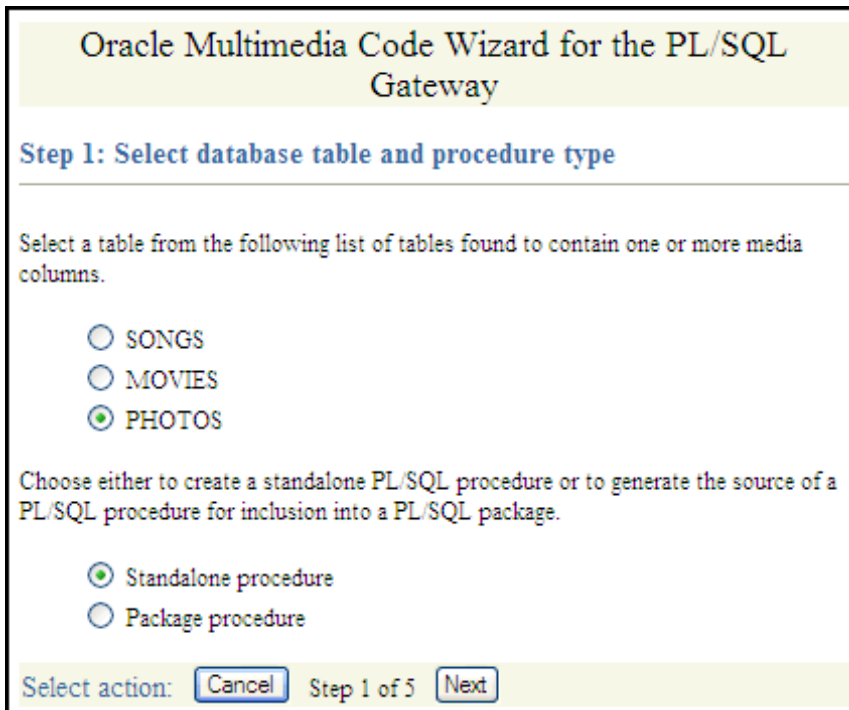


Select the required function, then click the **Next** button.

- Create media retrieval procedure
- Create media upload procedure

2. Select **PHOTOS** and **Standalone procedure** from **Step 1: Select database table and procedure type**, as shown in [Figure 4-6](#). Then, click **Next**.

Figure 4-6 Media Upload Step 1: Select Database Table and Procedure Type



Oracle Multimedia Code Wizard for the PL/SQL Gateway

Step 1: Select database table and procedure type

Select a table from the following list of tables found to contain one or more media columns.

- SONGS
- MOVIES
- PHOTOS

Choose either to create a standalone PL/SQL procedure or to generate the source of a PL/SQL procedure for inclusion into a PL/SQL package.

- Standalone procedure
- Package procedure

Select action: Step 1 of 5

3. Select **Use existing document table** from **Step 2: Select PL/SQL Gateway document upload table**, as shown in [Figure 4-7](#), because the `SCOTTGW` DAD is configured to use this document table. Then, click **Next**.

Figure 4-7 Media Upload Step 2: Select PL/SQL Gateway Document Upload Table

Oracle Multimedia Code Wizard for the PL/SQL Gateway

Step 2: Select PL/SQL Gateway document upload table

All files uploaded using the PL/SQL Gateway are uploaded into a document table. The media upload procedure created by this code wizard moves uploaded media from the specified document table to the application's table. To avoid transient files appearing temporarily in a document table used by another application component, choose a document table that is not being used to store documents permanently.

Note: Be sure to specify the selected document table in the application's Database Access Descriptor (DAD). If the DAD already specifies a different document table, create a new DAD for media uploads.

The document table currently specified for the `scottgw` DAD is: `WPG_DOCUMENT`

Choose either to select an existing document table or to create a new document table.

Use existing document table

Select an existing PL/SQL Gateway document table from the following list.

`WPG_DOCUMENT`

Create new document table

Enter a table name below to create a new document table.

Table name:

Select action: Step 2 of 5

4. Check **PHOTO (ORDIMAGE)**, select **ID (Primary key)**, and select **Conditional insert or update** from **Step 3: Select data access and media column(s)**, as shown in [Figure 4-8](#). Then, click **Next**.

Figure 4-8 Media Upload Step 3: Select Data Access and Media Column(s)

Oracle Multimedia Code Wizard for the PL/SQL Gateway

Step 3: Select data access and media column(s)

Select the column or columns to which media data is to be uploaded from the following list of media columns found in the PHOTOS table. If the table contains multiple media columns, you may select multiple columns to allow more than one media item to be uploaded from a single HTML form.

THUMBNAIL (ORDIMAGE)
 PHOTO (ORDIMAGE)

Select the column to be used to locate the media data from the following list of columns found in the PHOTOS table.

ID (Primary key)
 DESCRIPTION

Choose how the generated procedure will access the table to store uploaded media data. You may choose to insert a new row into the table, to update an existing row in the table, or to conditionally insert a new row if an existing row does not exist.

Insert new row
 Update existing row
 Conditional insert or update

Select action: Step 3 of 5

5. Check **DESCRIPTION**, accept the default procedure name, **UPLOAD_PHOTOS_PHOTO**, and select **Create procedure in the database** from **Step 4: Select additional columns and procedure name**, as shown in [Figure 4-9](#). Then, click **Next**.

Figure 4-9 Media Upload Step 4: Select Additional Columns and Procedure Name

The screenshot shows a wizard window titled "Oracle Multimedia Code Wizard for the PL/SQL Gateway". The current step is "Step 4: Select additional columns and procedure name". The window contains the following elements:

- Header:** Oracle Multimedia Code Wizard for the PL/SQL Gateway
- Step Title:** Step 4: Select additional columns and procedure name
- Text:** Optionally select any additional columns to be stored in the table along with the media data. The primary key and any columns with a unique or not-null constraint are selected automatically. If updating an existing row, simply clear any columns you do not wish to be stored. Note that the key column selected in the previous step is always included.
- Checkbox:** DESCRIPTION
- Text:** Choose a name for the media upload procedure. You can accept the default provided or supply a different name.
- Text Input:** Procedure name:
- Text:** Choose either to create the procedure in the database or to generate the procedure source code only. In either case you will subsequently have the opportunity to view the generated source code.
- Radio Buttons:**
 - Create procedure in the database
 - Generate procedure source only
- Footer:** Select action: Step 4 of 5

6. Review the options you selected from **Step 5: Review selected options**, as shown in [Figure 4-10](#). If the options selected are correct, click **Finish**.

Figure 4-10 Media Upload Step 5: Review Selected Options

Oracle Multimedia Code Wizard for the PL/SQL Gateway

Step 5: Review selected options

Click the **Finish** button if the following options are correct.

Procedure type:	Standalone
Table name:	PHOTOS
Media column(s):	PHOTO (ORDIMAGE)
Key column:	ID
Additional column(s):	DESCRIPTION
Table access mode:	Conditional update or insert
Procedure name:	UPLOAD_PHOTOS_PHOTO
Function:	Create procedure in the database

Select action: Step 5 of 5

7. The message `Procedure created successfully: UPLOAD_PHOTOS_PHOTO` is displayed on the **Compile procedure and review generated source** page, as shown in Figure 4-11.

Figure 4-11 Compiled Upload Procedure with Success Message

Oracle Multimedia Code Wizard for the PL/SQL Gateway

Compile procedure and review generated source

Procedure created successfully: UPLOAD_PHOTOS_PHOTO

Click the **View** button to display the compiled PL/SQL source code in a pop-up window. To save the source in a file for editing, select **Save As...** from your browser's **File** pull-down menu.

Click to display generated source:

Enter a DAD name, or accept the default provided, then click the **Test** button to display an HTML form in a pop-up window to upload media to the database to test the generated PL/SQL procedure. To save the source in a file for editing, select **Save As...** from your browser's **File** pull-down menu.

DAD:

Select action:

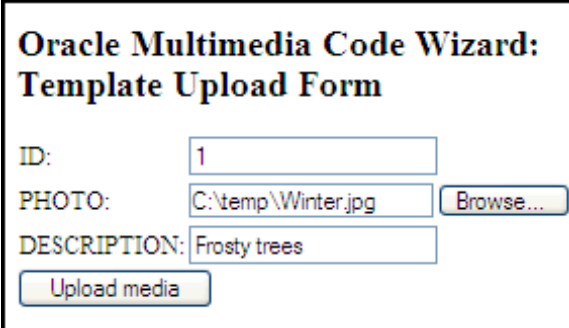
To review the compiled PL/SQL source code in another window, click **View**. Assuming you have configured the `SCOTT` DAD and specified `MEDIA_UPLOAD_TABLE` as the document table, in the **DAD:** field, the DAD name `scottcw` is displayed by default.

To test the PL/SQL procedure created, click **Test**.

The **Oracle Multimedia Code Wizard: Template Upload Form** is displayed in another window.

8. Enter the value `1` in the **ID** field on the **Oracle Multimedia Code Wizard: Template Upload Form** window. Click **Browse...** to find and select the image you want to upload in the **PHOTO** field, and enter a brief description of the image to be uploaded in the **DESCRIPTION** field, as shown in [Figure 4-12](#). Then, click **Upload media**.

Figure 4-12 Template Upload Form for the Code Wizard



The screenshot shows a window titled "Oracle Multimedia Code Wizard: Template Upload Form". It contains three input fields: "ID:" with the value "1", "PHOTO:" with the value "C:\temp\Winter.jpg" and a "Browse..." button, and "DESCRIPTION:" with the value "Frosty trees". At the bottom of the form is a button labeled "Upload media".

The image is uploaded into the table row, and this message is displayed:

Media uploaded successfully.

9. Return to the **Compile procedure and review generated source** page. If you are finished testing, click **Done** to return to the **Main menu** page.

Related Topics

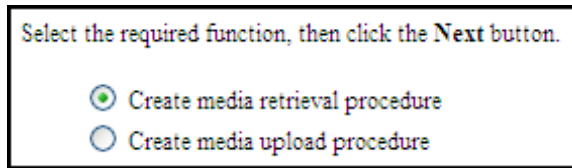
- [Sample Session: Using Images](#)
This sample session uses the `SCOTT` schema to demonstrate the creation of image media upload and retrieval procedures. To use a different schema, substitute a different schema name and password. Or, if you have changed the password for the `SCOTT` schema, use your new password.

4.2.5 Creating a Media Retrieval Procedure

Follow this process to create a media retrieval procedure using the Oracle Multimedia Code Wizard for the PL/SQL Gateway.

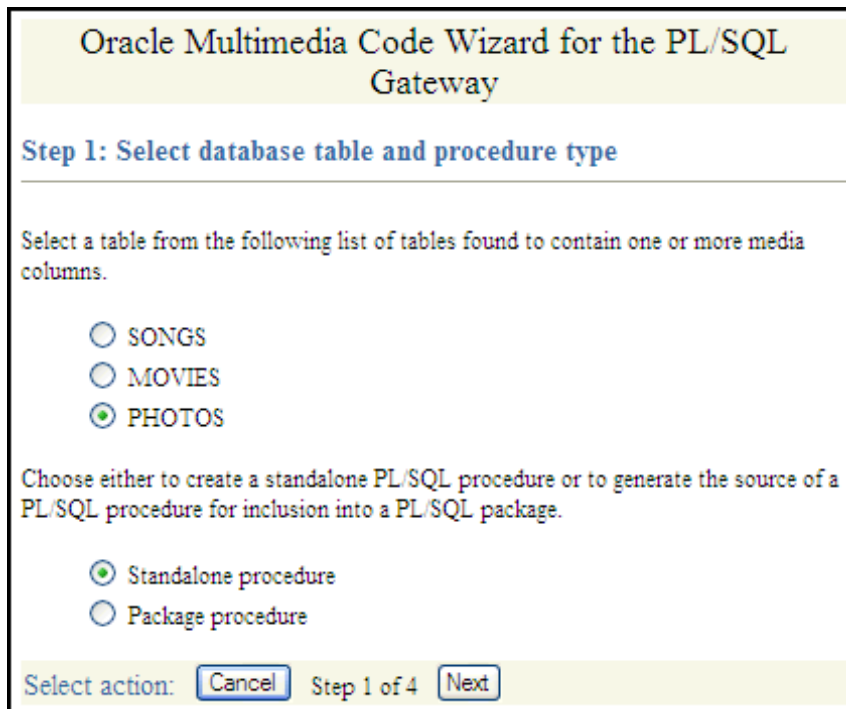
1. Select **Create media retrieval procedure** from the **Main menu** page, as shown in [Figure 4-13](#). Then, click **Next**.

Figure 4-13 Create a Media Retrieval Procedure



2. Select **PHOTOS** and **Standalone procedure** from **Step 1: Select database table and procedure type**, as shown in [Figure 4-14](#). Then, click **Next**.

Figure 4-14 Media Retrieval Step 1: Select Database Table and Procedure Type



3. Select **PHOTO (ORDIMAGE)** and **ID (Primary key)** from **Step 2: Select media column and key column**, as shown in [Figure 4-15](#). Then, click **Next**.

Figure 4-15 Media Retrieval Step 2: Select Media Column and Key Column

The screenshot shows a dialog box titled "Oracle Multimedia Code Wizard for the PL/SQL Gateway". The main heading is "Step 2: Select media column and key column". Below this, there are two instructions: "Select the column from which to retrieve the media data from the following list of media columns found in the PHOTOS table." and "Select the column to be used to locate the media data from the following list of columns found in the PHOTOS table." The first instruction has two radio button options: "THUMBNAIL (ORDIMAGE)" and "PHOTO (ORDIMAGE)", with "PHOTO (ORDIMAGE)" selected. The second instruction has two radio button options: "ID (Primary key)" and "ROWID (Unique)", with "ID (Primary key)" selected. At the bottom, there is a "Select action:" label followed by "Cancel", "Back", "Step 2 of 4", and "Next" buttons.

4. Accept the default procedure name, `GET_PHOTOS_PHOTO`, the default parameter name, `MEDIA_ID`, and **Create procedure in the database** from **Step 3: Select procedure name and parameter name**, as shown in [Figure 4-16](#). Then, click **Next**.

Figure 4-16 Media Retrieval Step 3: Select Procedure Name and Parameter Name

The screenshot shows the Oracle Multimedia Code Wizard for the PL/SQL Gateway. The title is "Oracle Multimedia Code Wizard for the PL/SQL Gateway". The step is "Step 3: Select procedure name and parameter name". The instructions say: "Choose a name for the media retrieval procedure. You can accept the default provided or supply a different name." The "Procedure name:" field contains "GET_PHOTOS_PHOTO". The instructions then say: "Choose a name for the parameter used to supply the key value. You can accept the default provided or supply a different name. The parameter name is used in a media retrieval URL as follows: http://host/DAD/proc-name?param-name=key-value". The "Parameter name:" field contains "MEDIA_ID". Below this, there are two radio button options: "Create procedure in the database" (which is selected) and "Generate procedure source only". At the bottom, there are buttons for "Cancel", "Back", "Step 3 of 4", and "Next".

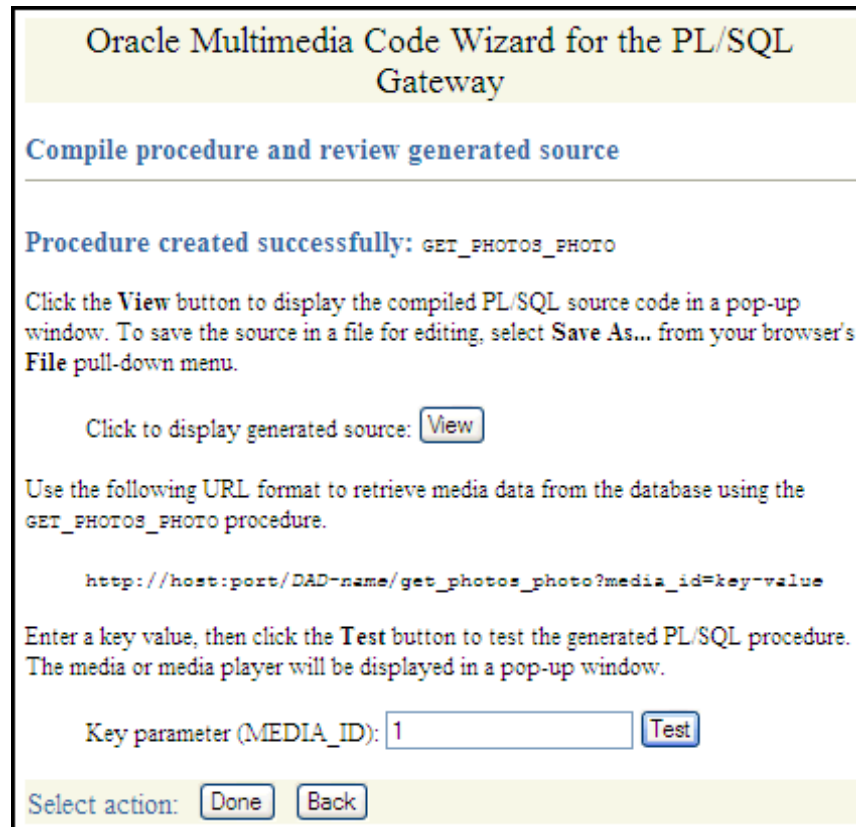
5. Review the options you selected from **Step 4: Review Selected Options**, as shown in [Figure 4-17](#). If the options selected are correct, click **Finish**.

Figure 4-17 Media Retrieval Step 4: Review Selected Options

The screenshot shows the Oracle Multimedia Code Wizard for the PL/SQL Gateway. The title is "Oracle Multimedia Code Wizard for the PL/SQL Gateway". The step is "Step 4: Review selected options". The instructions say: "Click the **Finish** button if the following options are correct." The summary shows: "Procedure type: Standalone", "Table name: PHOTOS", "Media column: PHOTO (ORDIMAGE)", "Key column: ID", "Procedure name: GET_PHOTOS_PHOTO", "Parameter name: MEDIA_ID", and "Function: Create procedure in the database". At the bottom, there are buttons for "Cancel", "Back", "Step 4 of 4", and "Finish".

- The message Procedure created successfully: GET_PHOTOS_PHOTO is displayed in the **Compile procedure and review generated source** page, as shown in Figure 4-18.

Figure 4-18 Compiled Retrieval Procedure with Success Message



To review the compiled PL/SQL source code in another window, click **View**.

To test the PL/SQL procedure created, assuming you have an image loaded in the database with an ID value of 1, enter the value 1 for the Key parameter (`MEDIA_ID`), then click **Test**.

The image is retrieved from the table row and displayed in another window.

- Click **Done** to return to the **Main menu** page.

Related Topics

- [Sample Session: Using Images](#)

This sample session uses the `SCOTT` schema to demonstrate the creation of image media upload and retrieval procedures. To use a different schema, substitute a different schema name and password. Or, if you have changed the password for the `SCOTT` schema, use your new password.

4.2.6 Using the PL/SQL Gateway Document Table

All files uploaded using the PL/SQL Gateway are stored in a document table. Media upload procedures created by the Code Wizard automatically move uploaded media

from the specified document table to the application's table. To avoid transient files from appearing temporarily in a document table used by another application component, use a document table that is not being used to store documents permanently.

Specify the selected document table in the application's database access descriptor (DAD). If the DAD specifies a different document table, create a new DAD for media upload procedures. If you choose to create a new document table, the Code Wizard creates a table with the following format:

```
CREATE TABLE document-table-name
  ( name          VARCHAR2(256) UNIQUE NOT NULL,
    mime_type     VARCHAR2(128),
    doc_size      NUMBER,
    dad_charset   VARCHAR2(128),
    last_updated  DATE,
    content_type  VARCHAR2(128),
    blob_content  BLOB )
--
-- store BLOBs as SecureFiles LOBs
--
LOB(blob_content) STORE AS SECUREFILE;
```



See Also:

Oracle Database Development Guide for more information about file upload and document tables

4.2.7 How Time Zone Information Is Used to Support Browser Caching

User response times are improved and network traffic is reduced if a browser can cache resources received from a Web server and subsequently use those cached resources to satisfy future requests. This section describes, at a very high level, how the browser caching mechanism works and how the Code Wizard utility package is used to support that mechanism. Within this discussion, all HTTP date and time stamps are expressed in Coordinated Universal Time (UTC).

All HTTP responses include a Date header, which indicates the date and time when the response was generated. When a Web server sends a resource in response to a request from a browser, it can also include the Last-Modified HTTP response header, which indicates the date and time when the requested resource was last modified. The Last-Modified header must not be *later* than the Date header.

After receiving and caching a resource, if a browser must retrieve the same resource again, it sends a request to the Web server with the If-Modified-Since request header specified as the value of the Last-Modified date, which was returned by the application server when the resource was previously retrieved and cached. When the Web server receives the request, it compares the date in the If-Modified-Since request header with the last update time of the resource. Assuming the resource still exists, if the resource has not changed since it was cached by the browser, the Web server responds with an HTTP 304 Not Modified status with no response body, which indicates that the browser can use the resource currently stored in its cache. Assuming again that the resource still exists, if the request does not include an If-Modified-Since header or if the

resource has been updated since it was cached by the browser, the Web server responds with an HTTP 200 OK status and sends the resource to the browser.

The `ORDImage`, `ORDAudio`, `ORDVideo`, and `ORDDoc` objects all possess an `updateTime` attribute stored as a `DATE` in the embedded `ORDSource` object. Although the `DATE` data type has no support for time zones or daylight savings time, Oracle Database does support time zones and also provides functions for converting a `DATE` value stored in a database to UTC.

When a response is first returned to a browser, a media retrieval procedure sets the Last-Modified HTTP response header based on the `updateTime` attribute. If a request for media data includes an If-Modified-Since header, the media retrieval procedure compares the value with the `updateTime` attribute and returns an appropriate response. If the resource in the browser's cache is still valid, an HTTP 304 Not Modified status is returned with no response body. If the resource has been updated since it was cached by the browser, then an HTTP 200 OK status is returned with the media resource as the response body. Media retrieval procedures created by the Code Wizard call the utility package to convert a `DATE` value stored in the database to UTC. The utility package uses the time zone information stored with Oracle Database and the date and time functions to convert database date and time stamps to UTC. To ensure that the resulting date conforms to the rule for the Last-Modified date described previously, the time zone information must be specified correctly.

See Also:

- *W3C HTTP- Hypertext Transfer Protocol* for more information about the HTTP specification
- *Oracle Database Administrator's Guide* for more information about how to set a time zone for a database
- *Oracle Database SQL Language Reference* for more information about date and time functions

4.3 Sample Session: Using Images

This sample session uses the `SCOTT` schema to demonstrate the creation of image media upload and retrieval procedures. To use a different schema, substitute a different schema name and password. Or, if you have changed the password for the `SCOTT` schema, use your new password.

Perform these steps:

Step 1. Create a table to store images for the application after connecting to a schema with privileges to create a table.

For example:

```
SQL> CREATE TABLE cw_images_table( id NUMBER PRIMARY KEY,
                                     description VARCHAR2(30) NOT NULL,
                                     location VARCHAR2(30),
                                     image ORDSYS.ORDIMAGE )
--
-- store media as SecureFiles LOBs
```

```
--  
LOB(image.source.localdata) STORE AS SECUREFILE;
```

Step 2. Create the `SCOTT` DAD to be used to create the procedures.

1. Set your Web browser to the Oracle HTTP Server Home page. Select **PL/SQL Properties** in the Administration page to open the `mod_plsql` Services page.
2. On the `mod_plsql` Services page, scroll to the DAD Status section. Then, click **Create** to open the DAD Type page.
3. Select the DAD type to be **General**. Then, click **Next** to open the Database Connection page.
4. Enter `/scottcw` in the DAD Name field. Enter `SCOTT` for the database account, and leave the password blank. Enter the connection information in the Database Connectivity Information section. Enter `ORDCWPKG.MENU` in the Default page field, and leave the other fields blank. Then, click **Next** to open the Document, Alias, and Session page.
5. On the Document, Alias, and Session page, enter `MEDIA_UPLOAD_TABLE` for the Document Table. Then, click **Apply**.
6. Restart Oracle HTTP Server for the changes to take effect.

Step 3. Authorize the use of the `SCOTT` DAD and `SCOTT` schema with the Code Wizard.

1. Enter the Code Wizard's administration URL into your browser's location bar, then enter the `ORDSYS` user name and password when prompted by the browser, for example:

```
http://<hostname>:<port-number>/ordcadmin
```

2. Select the DAD authorization function from the Code Wizard's **Main menu** and click **Next**. Enter the name of the demonstration DAD, `SCOTT`, and the user name `SCOTT`, then click **Apply**. Click **Done** when the confirmation window is displayed.

Step 4. Change DADs to the `SCOTT` DAD.

1. Click **Change DAD** from the Code Wizard's **Main menu**.
2. Click Change to `SCOTT`, if it is not already selected, then click **Next**.
3. Enter the user name `SCOTT` and the password for the user `SCOTT` when prompted for the user name and password, then click **OK**.

The **Main menu** now displays the current DAD as `SCOTT` and the current schema as `SCOTT`.

Step 5. Create and test the media upload procedure.

Click **Create media upload procedure** from the **Main menu**, then click **Next**.

1. Select the database table and procedure type.
 - a. Click the `CW_IMAGES_TABLE` database table.
 - b. Click **Standalone procedure**.
 - c. Click **Next**.
2. Select the PL/SQL document upload table.

If there are no document tables in the `SCOTT` schema, the Code Wizard displays a message indicating this situation. In this case, accept the default table name provided, `CW_SAMPLE_UPLOAD_TABLE`, then click **Next**.

If there are existing document tables, but the `CW_SAMPLE_UPLOAD_TABLE` is not among them, click **Create new document table**, accept the default table name provided, `CW_SAMPLE_UPLOAD_TABLE`, then click **Next**.

If the `CW_SAMPLE_UPLOAD_TABLE` document table already exists, ensure that the **Use existing document table** and the **CW_SAMPLE_UPLOAD_TABLE** options are selected. Click **Next**.

3. Select the data access and media columns.
 - a. Click **IMAGE (ORDIMAGE)**.
 - b. Click **ID (Primary key)**.
 - c. Click **Conditional insert or update**.
 - d. Click **Next**.
4. Select additional columns and procedure names.
 - a. Ensure that **DESCRIPTION** checkmarked because this column has a `NOT NULL` constraint. (The **LOCATION** column is not checkmarked by default as there are no constraints on this column.)
 - b. Accept the procedure name provided, `UPLOAD_CW_IMAGES_TABLE_IMAGE`.
 - c. Click **Create procedure in the database**.
 - d. Click **Next**.
5. Review the following selected procedure creation options that are displayed:

Procedure type:	Standalone
Table name:	<code>CW_IMAGES_TABLE</code>
Media column(s):	<code>IMAGE (ORDIMAGE)</code>
Key column:	<code>ID</code>
Additional column(s):	<code>DESCRIPTION</code>
Table access mode:	Conditional update or insert
Procedure name:	<code>UPLOAD_CW_IMAGES_TABLE_IMAGE</code>
Function:	Create procedure in the database

Click **Finish**.

6. Compile the procedure and review the generated source information.

The Code Wizard displays this message:

```
Procedure created successfully: UPLOAD_CW_IMAGES_TABLE_IMAGE
```

- a. At the option **Click to display generated source:**, click **View** to view the generated source in another window. A copy of the generated source is shown at the end of Step 5, substep 6g.
- b. Close the window after looking at the generated source.
- c. Accept the **DAD:** name provided, `SCOTT`, then click **Test** to produce another window that displays a template file upload form that you can use to test the generated procedure.

- d. To customize the template file upload form, select **Save As...** from your browser's **File** menu to save the HTML source for editing.
- e. To test the template upload form, enter this information:
 - For the **ID:** column, enter the number 1 as the row's primary key.
 - For the **IMAGE** column, click **Browse...** and choose an image file to upload to the database.
 - For the **DESCRIPTION** column, enter a brief description of the image.
 - Click **Upload media**.

The Code Wizard displays a template completion window with the heading **Oracle Multimedia Code Wizard: Template Upload Procedure**, and, if the procedure is successful, the message: `Media uploaded successfully`.

- f. Close the window.
- g. Click **Done** on the **Compile procedure and review generated source** window to return to the **Main menu** of the Code Wizard.

A copy of the generated image upload procedure follows:

```
CREATE OR REPLACE PROCEDURE UPLOAD_CW_IMAGES_TABLE_IMAGE
( in_ID IN VARCHAR2,
  in_IMAGE IN VARCHAR2 DEFAULT NULL,
  in_DESCRIPTION IN VARCHAR2 DEFAULT NULL )
AS
  local_IMAGE ORDSYS.ORDIMAGE := ORDSYS.ORDIMAGE();
  local_ID CW_IMAGES_TABLE.ID%TYPE := NULL;
  upload_size INTEGER;
  upload_mimetype VARCHAR2( 128 );
  upload_blob BLOB;
BEGIN
  --
  -- Update the existing row.
  --
  UPDATE CW_IMAGES_TABLE mtbl
    SET mtbl.IMAGE = local_IMAGE,
        mtbl.DESCRPTION = in_DESCRIPTION
    WHERE mtbl.ID = in_ID
    RETURN mtbl.ID INTO local_ID;
  --
  -- Conditionally insert a new row if no existing row is updated.
  --
  IF local_ID IS NULL
  THEN
    --
    -- Insert the new row into the table.
    --
    INSERT INTO CW_IMAGES_TABLE ( ID, IMAGE, DESCRIPTION )
      VALUES ( in_ID, local_IMAGE, in_DESCRIPTION );
  END IF;
  --
  -- Select Oracle Multimedia object(s) for update.
  --
  SELECT mtbl.IMAGE INTO local_IMAGE
    FROM CW_IMAGES_TABLE mtbl WHERE mtbl.ID = in_ID FOR UPDATE;
  --
```

```

-- Store media data for the column in_IMAGE.
--
IF in_IMAGE IS NOT NULL
THEN
  SELECT dtbl.doc_size, dtbl.mime_type, dtbl.blob_content INTO
         upload_size, upload_mimetype, upload_blob
  FROM CW_IMAGE_UPLOAD_TABLE dtbl WHERE dtbl.name = in_IMAGE;
IF upload_size > 0
THEN
  dbms_lob.copy( local_IMAGE.source.localData,
                 upload_blob,
                 upload_size );
  local_IMAGE.setLocal();
BEGIN
  local_IMAGE.setProperties();
EXCEPTION
  WHEN OTHERS THEN
    local_IMAGE.contentLength := upload_size;
    local_IMAGE.mimeType := upload_mimetype;
  END;
END IF;
DELETE FROM CW_IMAGE_UPLOAD_TABLE dtbl WHERE dtbl.name = in_IMAGE;
END IF;
--
-- Update Oracle Multimedia objects in the table.
--
UPDATE CW_IMAGES_TABLE mtbl
  SET mtbl.IMAGE = local_IMAGE
  WHERE mtbl.ID = in_ID;
--
-- Display the template completion message.
--
http.print( '<html>' );
http.print( '<title>Oracle Multimedia Code Wizard: Template Upload
Procedure</title>' );
http.print( '<body>' );
http.print( '<h2> Oracle Multimedia Code Wizard:
Template Upload Procedure</h2>' );
http.print( 'Media uploaded successfully.' );
http.print( '</body>' );
http.print( '</html>' );
END UPLOAD_CW_IMAGES_TABLE_IMAGE;

```

The previous image upload procedure declares these input parameters and variables:

- a. In the declaration section, the procedure declares three input parameters: `in_ID`, `in_IMAGE`, and `in_DESCRIPTION`, then initializes the latter two to `NULL`.
- b. In the subprogram section, the following variables are declared:
 - The variable `local_IMAGE` is assigned the data type `ORDSYS.ORDIMAGE` and initialized with an empty BLOB using the `ORDIMAGE()` constructor.
 - The variable `local_ID` takes the same data type as the `ID` column in the table `CW_IMAGES_TABLE` and is initialized to `NULL`.

- Three additional variables are declared `upload_size`, `upload_mimetype`, and `upload_blob`, which are later given values from comparable column names `doc_size`, `mime_type`, and `blob_content` from the document table `CW_IMAGE_UPLOAD_TABLE`, using a `SELECT` statement in preparation for copying the content of the image BLOB data to the `ORDSYS.ORDIMAGE.source.localData` attribute.

Within the outer `BEGIN...END` executable statement section, the following operations are executed:

- a. Update the existing row in the table `CW_IMAGES_TABLE` for the `IMAGE` and `DESCRIPTION` columns and return the value of `local_ID` where the value of the `ID` column is the value of the `in_ID` input parameter.
- b. If the value returned of `local_ID` is `NULL`, conditionally insert a new row into the table `CW_IMAGES_TABLE` and initialize the instance of the `ORDImage` object type in the `image` column with an empty BLOB.
- c. Select the `ORDImage` object column `IMAGE` in the table `CW_IMAGES_TABLE` for update where the value of the `ID` column is the value of the `in_ID` input parameter.
- d. Select a row for the `doc_size`, `mime_type`, and `blob_content` columns from the document table and pass the values to the `upload_size`, `upload_mimetype`, and `upload_blob` variables where the value of the document table `Name` column is the value of the `in_IMAGE` input parameter.
- e. Perform a `DBMS_LOB` copy of the BLOB data from the table `CW_IMAGE_UPLOAD_TABLE` into the `ORDSYS.ORDIMAGE.source.localData` attribute, then call the `setLocal()` method to indicate that the image data is stored locally in the BLOB, and `ORDImage` methods are to look for corresponding data in the `source.localData` attribute.
- f. In the inner executable block, call the `ORDImage` `setProperties()` method to read the image data to get the values of the object attributes and store them in the `image` object attributes for the `ORDImage` object.
- g. If the `setProperties()` call fails, catch the exception and call the `contentLength()` method to get the size of the image and call the `mimeType()` method to get the MIME type of the image.
- h. Delete the row of data from the document table `CW_IMAGE_UPLOAD_TABLE` that was copied to the row in the table `CW_IMAGES_TABLE` where the value of the `Name` column is the value of the `in_IMAGE` input parameter.
- i. Update the `ORDImage` object `IMAGE` column in the table `CW_IMAGES_TABLE` with the content of the variable `local_IMAGE` where the value of the `ID` column is the value of the `in_ID` input parameter.
- j. Display a completion message on the HTML page to indicate that the media uploaded successfully using the `http.print` function from the PL/SQL Web Toolkit.

Step 6. Create and test a media retrieval.

Select **Create media retrieval procedure** from the **Main menu**, then click **Next**.

1. Select the database table and procedure type.

- a. Click **CW_IMAGES_TABLE**.
 - b. Click **Standalone procedure**.
 - c. Click **Next**.
2. Select the media column and key column.
 - a. Click **IMAGE (ORDIMAGE)**.
 - b. Click **ID (Primary key)**.
 - c. Click **Next**.
 3. Select the procedure name and parameter name.
 - a. Accept the procedure name provided, `GET_CW_IMAGES_TABLE_IMAGE`.
 - b. Accept the parameter name provided, `MEDIA_ID`.
 - c. Click **Create procedure in the database**.
 - d. Click **Next**.
 4. Review the following selected procedure creation options:
 5.

Procedure type:	Standalone
Table name:	CW_IMAGES_TABLE
Media column(s):	IMAGE (ORDIMAGE)
Key column:	ID
Procedure name:	GET_CW_IMAGES_TABLE_IMAGE
Parameter Name:	MEDIA_ID
Function:	Create procedure in the database

Click **Next**.

6. Compile the procedure and review the generated source.
The Code Wizard displays this message:

Procedure created successfully: `GET_CW_IMAGES_TABLE_IMAGE`

- a. Click **View** to view the generated source in another window. Close the window after looking at the generated source. A copy of the generated source is shown at the end of Step 6, substep 5e.
- b. Review the URL format used to retrieve images using the `GET_CW_IMAGES_TABLE_IMAGE` procedure.
- c. Enter the number 1 as the Key parameter, then click **Test** to test the procedure by retrieving the image uploaded previously.
The retrieved image is displayed in another window.
- d. Close the window.
- e. Click **Done** to return to the **Main menu**.

A copy of the generated image retrieval procedure follows:

```
CREATE OR REPLACE PROCEDURE GET_CW_IMAGES_TABLE_IMAGE (
  MEDIA_ID IN VARCHAR2 )
AS
  localObject ORDSYS.ORDIMAGE;
  localBlob BLOB;
```

```
localBfile BFILE;
HttpStatus NUMBER;
lastModDate VARCHAR2(256);
BEGIN
--
-- Retrieve the object from the database into a local object.
--
BEGIN
  SELECT mtbl.IMAGE INTO localObject FROM CW_IMAGES_TABLE mtbl
     WHERE mtbl.ID = MEDIA_ID;
EXCEPTION
  WHEN NO_DATA_FOUND THEN
    ordplsgwyutil.resource_not_found( 'MEDIA_ID', MEDIA_ID );
    RETURN;
END;

--
-- Check the update time if the browser sent an If-Modified-Since header.
--
IF ordplsgwyutil.cache_is_valid( localObject.getUpdateTime() )
THEN
  owa_util.status_line( ordplsgwyutil.http_status_not_modified );
  RETURN;
END IF;

--
-- Figure out where the image is.
--
IF localObject.isLocal() THEN
  --
  -- Data is stored locally in the localData BLOB attribute.
  --
  localBlob := localObject.getContent();
  owa_util.mime_header( localObject.getMimeType(), FALSE );
  ordplsgwyutil.set_last_modified( localObject.getUpdateTime() );
  owa_util.http_header_close();
  IF owa_util.get_cgi_env( 'REQUEST_METHOD' ) &lt;&gt; 'HEAD' THEN
    wpg_docload.download_file( localBlob );
  END IF;
ELSIF UPPER( localObject.getSourceType() ) = 'FILE' THEN

  --
  -- Data is stored as a file from which ORDSOURCE creates
  -- a BFILE.
  --
  localBfile := localObject.getBFILE();
  owa_util.mime_header( localObject.getMimeType(), FALSE );
  ordplsgwyutil.set_last_modified( localObject.getUpdateTime() );
  owa_util.http_header_close();
  IF owa_util.get_cgi_env( 'REQUEST_METHOD' ) &lt;&gt; 'HEAD' THEN
    wpg_docload.download_file( localBfile );
  END IF;

ELSIF UPPER( localObject.getSourceType() ) = 'HTTP' THEN --
  -- The image is referenced as an HTTP entity, so we have to
  -- redirect the client to the URL that ORDSOURCE provides.
  --
  owa_util.redirect_url( localObject.getSource() );
```

```
ELSE
  --
  -- The image is stored in an application-specific data
  -- source type for which no default action is available.
  --
  NULL;
END IF;
END GET_CW_IMAGES_TABLE_IMAGE;
```

The image retrieval procedure shown previously declares these input parameters and variables:

- a. In the declaration section, the procedure declares one input parameter: `MEDIA_ID`.
- b. In the subprogram section, the following variables are declared:
 - The variable `localObject` is assigned the data type `ORDSYS.ORDIMAGE`.
 - The variable `localBlob` is a BLOB data type, the variable `localBfile` is a BFILE data type, `httpStatus` is a NUMBER, and `lastModDate` is a VARCHAR2 with a maximum size of 256 characters.

Within the outer BEGIN...END executable statement section, the following operations are executed:

- a. Select the ORDImage object column `IMAGE` in the table `CW_IMAGES_TABLE` where the value of the `ID` column is the value of the `MEDIA_ID` input parameter.
- b. In the inner executable block, when no data is found, raise an exception and call the `resource_not_found` function of the PL/SQL Gateway and get the value of the `MEDIA_ID` input parameter.
- c. Check the update time if the browser sent an If-Modified-Since header by calling the `getUpdateTime()` method passed into the `cache_is_valid` function of the PL/SQL Gateway.
- d. If the cache is valid, send an HTTP status code to the client using the PL/SQL Web Toolkit `owa_util` package `status_line` procedure passing in the call to the `http_status_not_modified` function.
- e. Determine where the image data is stored; call the ORDImage `isLocal()` method, which returns a Boolean expression of true if the image data is stored locally in the BLOB, then get the handle to the local BLOB.
 - If the value is true, assign the variable `localBlob` the ORDImage `getContent()` method to get the handle of the local BLOB containing the image data.
 - Call the ORDImage `getMimeType()` method to determine the image's MIME type and pass this to the `owa_util.mime_header` procedure and keep the HTTP header open.
 - Call the ORDImage `getUpdateTime()` method to get the time the image was last modified and pass this to the `ordplsgwyutil.set_last_modified` procedure.

- Close the HTTP header by calling the `owa_util.http_header_close()` procedure.
 - Call the `owa_util.get_cgi_env` procedure and if the value of the request method is not `HEAD`, then use the `wpg_docload.download_file` procedure to pass in the value of `localBlob` that contains the LOB locator of the BLOB containing the image data to download the image from the database.
- f. If the `ORDImage.isLocal()` method returns false, call the `ORDImage.getSourceType()` method to determine if the value is `FILE`; if so, then the image data is stored as an external file on the local file system. Then, get the LOB locator of the BFILE containing the image data.
- Assign the variable `localBfile` the `ORDImage.getBfile()` method to get the LOB locator of the BFILE containing the image data.
 - Call the `ORDImage.getMimeType()` method to determine the image's MIME type and pass this to the `owa_util.mime_header` procedure and keep the HTTP header open.
 - Call the `ORDImage.getUpdateTime()` method to get the time the image was last modified and pass this to the `ordplsgwyutil.set_last_modified` procedure.
 - Close the HTTP header by calling the `owa_util.http_header_close()` procedure.
 - Call the `owa_util.get_cgi_env` procedure and if the value of the request method is not `HEAD`, then use the `wpg_docload.download_file` procedure to pass in the value of `localBfile` that contains the LOB locator of the BFILE containing the image data to download the image from the file.
- g. If the `ORDImage.isLocal()` method returns false, call the `ORDImage.getSourceType()` method to determine if the value is `HTTP`; if so, then the image data is stored at an HTTP URL location, which then redirects the client to the URL that `ORDSource` provides using the `owa_util.redirect_url` procedure.
- h. If the `ORDImage.isLocal()` method returns false, call the `ORDImage.getSourceType()` method to determine if the value is `FILE` or `HTTP`; if it is neither, then the image is stored in an application-specific data source type that is not recognized or supported by Oracle Multimedia.

 **See Also:**

Oracle Database Security Guide for more information about creating secure passwords

4.4 Known Restrictions of the Oracle Multimedia Code Wizard

The Code Wizard supports the features of Oracle Multimedia, with a few known restrictions.

The following restrictions are known for the Oracle Multimedia Code Wizard:

- Tables with composite primary keys are not supported.

To use a table with a composite primary key, create an upload or download procedure, then edit the generated source to support all the primary key columns. For example, for a media retrieval procedure, this might involve adding an additional parameter, then specifying that parameter in the `where` clause of the `SELECT` statement.

- User object types containing embedded Oracle Multimedia object types are not recognized by the Oracle Multimedia Code Wizard.

5

Working with Metadata in Oracle Multimedia Images

Image files can contain information about the content of the images, the image pixel data, and image metadata.

In general, data about data is referred to as [metadata](#). In this case, metadata refers to additional information about the actual images, which is stored in the image files along with the images.

This chapter includes these sections:

- [Metadata Concepts](#)
- [Oracle Multimedia Image Metadata Concepts](#)
- [Image File Formats](#)
- [Image Metadata Formats](#)
- [Representing Metadata Outside Images](#)
- [Oracle Multimedia Image Metadata Examples](#)
- [Metadata References](#)

5.1 Metadata Concepts

Several types of metadata can be stored in an image file, and each type can serve a different purpose.

One type, [technical metadata](#), is used to describe an image in a technical sense. For example, technical metadata can include attributes about an image, such as its height and width, in pixels, or the type of compression used to store it.

Another type, [content metadata](#), can further describe the content of an image, the name of the photographer, and the date and time when a photograph was taken.

Metadata is stored in image files using a variety of mechanisms. Digital cameras and scanners automatically insert metadata into the images they create. Digital photograph processing applications like Adobe Photoshop enable users to add or edit metadata to be stored with the image. Annotating digital images with additional metadata is a common practice in photographic and news gathering applications, for image archiving usages, and at the consumer level.

Storing metadata with image data in the same containing file provides encapsulation. With encapsulation, both types of data can be shared and exchanged reliably as one unit. Metadata that is stored in the image file format is referred to as [embedded metadata](#).

5.2 Oracle Multimedia Image Metadata Concepts

For a number of image file formats, Oracle Multimedia can extract and manage a limited set of metadata attributes. These attributes include: height, width, contentLength, fileFormat, contentFormat, compressionFormat, and mimeType. For a limited number of image file formats, Oracle Multimedia can extract a rich set of metadata attributes. This metadata is represented in schema-based XML documents. These XML documents can be stored in a database, indexed, searched, updated, and made available to applications using the standard mechanisms of Oracle Database.

Oracle Multimedia can also write or embed metadata supplied by users into a limited number of image file formats. The application provides the metadata as a schema-based XML document. Oracle Multimedia processes the XML document and writes the metadata into the image file.

5.3 Image File Formats

Oracle Multimedia supports metadata extraction and metadata embedding for the GIF, TIFF, and JPEG file formats.



See Also:

Oracle Multimedia Reference for information about the image file formats supported by Oracle Multimedia

5.4 Image Metadata Formats

The term image metadata format refers to the standard protocols and techniques used to store image metadata within an image file.

The following subsections describe the embedded image metadata formats supported by Oracle Multimedia:

- [EXIF](#)
- [IPTC-IIM](#)
- [XMP](#)

5.4.1 EXIF

Oracle Multimedia supports the extraction of EXIF metadata from TIFF and JPEG file formats.

The Exchangeable Image File Format (EXIF) is the standard for image file storage for digital still cameras. It was developed by the Japan Electronic Industry Development Association (JEIDA) as a standard way of storing images created by digital cameras and metadata about the images. EXIF image metadata can be stored in TIFF and JPEG format images.

5.4.2 IPTC–IIM

Oracle Multimedia supports the extraction of IPTC metadata from TIFF and JPEG file formats.

The International Press Telecommunications Council-Information Interchange Model (IPTC-IIM) Version 4 is a standard developed jointly by the International Press Telecommunications Council and the Newspaper Association of America. This metadata standard is designed to capture information that is important to the activities of news gathering, reporting, and publishing. These information records are commonly referred to as IPTC tags.

The use of embedded IPTC tags in image file formats became widespread with the use of the Adobe Photoshop tool for image editing. IPTC metadata can be stored in TIFF and JPEG format images.

5.4.3 XMP

Oracle Multimedia supports the extraction of XMP metadata from GIF, TIFF, and JPEG file formats. Oracle Multimedia also supports writing XMP data packets into GIF, TIFF, and JPEG file formats.

The Extensible Metadata Platform (XMP) is a standard metadata format, developed by Adobe, for the creation, processing, and interchange of metadata in a variety of applications. XMP uses Resource Description Framework (RDF) technology for data modeling. XMP also defines how the data model is serialized (converted to a byte stream), and embedded within an image file.

5.5 Representing Metadata Outside Images

After metadata has been extracted from the binary image file, the next step is to represent the metadata in a form that can be easily stored, indexed, queried, updated, and presented.

Oracle Multimedia returns image metadata in XML documents. These documents are based on XML schemas that Oracle Multimedia registers with the database. Each type of image metadata has a separate XML schema. These XML schemas are used by the metadata procedures and methods of the `ORD_IMAGE` PL/SQL package and the `ORDImage` object type, respectively.

The XML documents can be stored in `XMLType` columns within the database. These documents are easily searched and processed using the wide range of standards-based XML technologies provided by Oracle XML DB.

See Also:

- *Oracle Multimedia Reference* for complete definitions of the XML schemas supported by Oracle Multimedia
- *Oracle XML DB Developer's Guide* for more information about the XML technologies provided by Oracle XML DB

5.6 Oracle Multimedia Image Metadata Examples

This topic introduces a set of examples that show how to extract and embed image metadata using a table from the Oracle Multimedia PL/SQL Web Toolkit Photo Album sample application.

The following examples of metadata extraction and embedding use the `photos` table, which is defined by the Photo Album sample application. The implementation of the Photo Album sample application is defined in the PL/SQL package `PHOTO_ALBUM`.

The `photos` table stores two instances of an image: the full-size photograph and a thumbnail image. This table can also store up to four different image metadata documents. These documents are stored in the columns named `metaORDImage`, `metaEXIF`, `metaIPTC`, and `metaXMP`, and represent image metadata from the `ORDImage`, `EXIF`, `IPTC`, and `XMP` metadata formats, respectively. The metadata columns are of type `XMLType`, and they are bound to the corresponding metadata XML schemas that Oracle Multimedia provides.

The following subsections describe some operations you can perform with image metadata using the `ORDImage` object type:

- [Creating a Table for Metadata Storage](#)
- [Extracting Image Metadata](#)
- [Embedding Image Metadata](#)



Note:

These examples could have used BLOBs to store the image and thumbnail image, and the `ORD_IMAGE` PL/SQL package instead of `ORDImage` object methods.

Related Topics

- [Oracle Multimedia PL/SQL Photo Album Sample Application](#)
The Oracle Multimedia PL/SQL Web Toolkit Photo Album sample application is a media upload and retrieval Web application using Oracle Multimedia object types. This application uses the PL/SQL Gateway and PL/SQL Web Toolkit.

5.6.1 Creating a Table for Metadata Storage

Before you can extract or embed metadata, you must create the table and columns where the metadata is to be stored.

The following PL/SQL code segment creates the `photos` table with four `XMLTYPE` columns (`metaORDImage`, `metaEXIF`, `metaIPTC`, and `metaXMP`) to store each type of image metadata, and two `ORDIMAGE` columns (`image` and `thumb`) for the original image and the thumbnail image, respectively. Although this example shows the use of the `ORDImage` object type, the columns `image` and `thumb` could be of type `BLOB` rather than `ORDImage`.

Each metadata column is bound to its corresponding metadata schema. For example, the column `metaEXIF` is bound to the XML schema with namespace `http://xmlns.oracle.com/ord/meta/exif`, and is defined as the XML element `exifMetadata`.

The code statements where the image metadata columns are defined and bound to XML schemas are highlighted in bold.

```
--
-- Create the PHOTOS table
--
CREATE TABLE photos( id          NUMBER PRIMARY KEY,
                     description  VARCHAR2(40) NOT NULL,
                     metaORDImage XMLTYPE,
                     metaEXIF    XMLTYPE,
                     metaIPTC    XMLTYPE,
                     metaXMP     XMLTYPE,
                     image        ORDSYS.ORDIMAGE,
                     thumb        ORDSYS.ORDIMAGE )

--
-- store full-size images and thumbnail images as SecureFiles LOBs
--
LOB(image.source.localdata) STORE AS SECUREFILE
LOB(thumb.source.localdata) STORE AS SECUREFILE
-- and bind the XMLType columns to the Oracle Multimedia metadata schemas
XMLType COLUMN metaORDImage
  XMLSCHEMA "http://xmlns.oracle.com/ord/meta/ordimage"
  ELEMENT "ordImageAttributes"
XMLType COLUMN metaEXIF
  XMLSCHEMA "http://xmlns.oracle.com/ord/meta/exif"
  ELEMENT "exifMetadata"
XMLType COLUMN metaIPTC
  XMLSCHEMA "http://xmlns.oracle.com/ord/meta/iptc"
  ELEMENT "iptcMetadata"
XMLType COLUMN metaXMP
  XMLSCHEMA "http://xmlns.oracle.com/ord/meta/xmp"
  ELEMENT "xmpMetadata";
```

5.6.2 Extracting Image Metadata

This topic uses an example to show how to extract image metadata.

The following PL/SQL procedure extracts metadata from an image and stores it in the specified columns in the `photos` table you created. This procedure demonstrates the `getMetadata()` method, which returns an array of XML documents. The root element of each document is examined to determine the metadata type. The `UPDATE` statement stores the documents in the corresponding columns in the `photos` table.

The code statement where the `getMetadata()` method is called is highlighted in bold.

```
--
-- fetch the metadata and sort the results
--
PROCEDURE extractMetadata(inID IN INTEGER)
IS
  img ORDSYS.ORDIMAGE;
  metav XMLSequenceType;
  meta_root VARCHAR2(40);
  xmlORD XMLType;
  xmlXMP XMLType;
  xmlEXIF XMLType;
```

```

    xmlIPTC XMLType;

BEGIN

-- select the image
SELECT image
INTO img
FROM PHOTOS
WHERE id = inID;

-- extract all the metadata
metav := img.getMetadata( 'ALL' );

-- process the result array to discover what types of metadata were
returned
FOR i IN 1..metav.count() LOOP
    meta_root := metav(i).getRootElement();
    CASE meta_root
        WHEN 'ordImageAttributes' THEN xmlORD := metav(i);
        WHEN 'xmpMetadata' THEN xmlXMP := metav(i);
        WHEN 'iptcMetadata' THEN xmlIPTC := metav(i);
        WHEN 'exifMetadata' THEN xmlEXIF := metav(i);
        ELSE NULL;
    END CASE;
END LOOP;

-- Update metadata columns
--
UPDATE photos
SET metaORDImage = xmlORD,
    metaEXIF = xmlEXIF,
    metaIPTC = xmlIPTC,
    metaXMP = xmlXMP
WHERE id = inID;

END extractMetadata;

```

5.6.3 Embedding Image Metadata

This topic uses an example to show how to embed image metadata.

The following PL/SQL procedure demonstrates the `putMetadata()` method. This procedure accepts six arguments. The `entry_id` argument identifies the image in the `photos` table to be updated. The remaining arguments (`title`, `creator`, `date`, `description`, and `copyright`) are strings to be formatted into an XMP packet and embedded within the target image.

This example creates an XML document instance based on the Oracle Multimedia XML schema for XMP metadata. (This schema is preregistered with Oracle XML DB.) The schema for XMP metadata defines a single, global element `<xmpMetadata>`. The `<xmpMetadata>` element contains a single, well-formed RDF document. The RDF document contains a single `<RDF>` element, which is derived from the `rdf` namespace. This RDF document is constructed using elements defined by the Dublin Core schema.

The call to the `putMetadata()` method embeds the metadata document into the image file. The `UPDATE` statement stores the new image and the new metadata back in the `photos` table.

The code statement where the `putMetadata()` method is called is highlighted in bold.

```
--
-- write the metadata to the image
--
PROCEDURE write_metadata( entry_id IN VARCHAR2,
                        title IN VARCHAR2,
                        creator IN VARCHAR2,
                        date IN VARCHAR2,
                        description IN VARCHAR2,
                        copyright IN VARCHAR2 )
IS
  img ORDSYS.ORDImage;
  xmp XMLType;
  buf VARCHAR2(5000);
BEGIN
  -- select the image
  SELECT image
  INTO img
  FROM PHOTOS
  WHERE id = entry_id FOR UPDATE;

  -- Create the XMP packet it must be schema valid
  -- to "http://xmlns.oracle.com/ord/meta/xmp"
  -- and contain an <RDF> element. This example uses
  -- the Dublin Core schema.

  /* An example XML instance document

  <xmpMetadata xmlns="http://xmlns.oracle.com/ord/meta/xmp"
              xsi:schemaLocation="http://xmlns.oracle.com/ord/meta/xmp
              http://xmlns.oracle.com/ord/meta/xmp"
              xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">

    <rdf:RDF xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#">
      <rdf:Description about="" xmlns:dc="http://purl.org/dc/elements/1.1/">
        <dc:title>A Winter Day</dc:title>
        <dc:creator>Frosty S. Man</dc:creator>
        <dc:date>21-Dec-2004</dc:date>
        <dc:description>a sleigh ride</dc:description>
        <dc:copyright>North Pole Inc.</dc:copyright>
      </rdf:Description>
    </rdf:RDF>
  </xmpMetadata>

  */

  buf := '<xmpMetadata xmlns="http://xmlns.oracle.com/ord/meta/xmp"
              xsi:schemaLocation="http://xmlns.oracle.com/ord/meta/xmp
              http://xmlns.oracle.com/ord/meta/xmp"
              xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" >
              <rdf:RDF xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#">
              <rdf:Description about="" xmlns:dc="http://purl.org/dc/elements/1.1/">';

  IF title IS NOT NULL THEN
    buf := buf || '<dc:title>' || htf.escape_sc(title) || '</dc:title>';
  END IF;

  IF creator IS NOT NULL THEN
    buf := buf || '<dc:creator>' || htf.escape_sc(creator)
    || '</dc:creator>';
  END IF;

```

```

END IF;
IF date IS NOT NULL THEN
  buf := buf || '<dc:date>' || htf.escape_sc(date)
         || '</dc:date>';
END IF;
IF description IS NOT NULL THEN
  buf := buf || '<dc:description>' || htf.escape_sc(description)
         || '</dc:description>';
END IF;
IF copyright IS NOT NULL THEN
  buf := buf || '<dc:copyright>' || htf.escape_sc(copyright)
         || '</dc:copyright>';
END IF;

buf := buf || '
</rdf:Description>
</rdf:RDF>
</xmpMetadata>';

-- create the XML document
xmp := XMLType.createXML(buf, 'http://xmlns.oracle.com/ord/meta/xmp');

-- write the metadata
img.putMetadata( xmp, 'XMP' );

-- update the image
UPDATE photos
SET image = img,
    metaXMP = xmp
WHERE id = entry_id;

END write_Metadata;

```

5.7 Metadata References

This topic includes a list of references for more information about working with image metadata.

The following Web sites provide information about standards and technologies related to working with metadata in images.

- Dublin Core, a standard schema for Dublin core elements
Dublin Core Metadata Initiative
- Extensible Metadata Platform (XMP), an open source file labeling technology for embedding metadata into files
Extensible Metadata Platform
- Resource Description Framework (RDF), a standard model for interchanging data on the Web (See *RDF Primer*)
Resource Description Framework

6

Oracle Multimedia Tuning Tips for DBAs

This chapter provides information and advice for Oracle DBAs who want to achieve more efficient storage and management of multimedia data in the database when using Oracle Multimedia.

The goals of your Oracle Multimedia application determine the resource requirements and how to allocate those resources. Because application development and design decisions have the greatest effect on performance, standard tuning methods must be applied to the system planning, design, and development phases of the project to achieve optimal results for your Oracle Multimedia application in a production environment.

Multimedia data consists of a variety of media types including images, audio clips, video clips, line drawings, and so on. All these media types are typically stored in LOBs. LOBs can be either internal BLOBs (stored in an internal database tablespace) or BFILEs (external LOBs in operating system files outside of the database tablespaces). This chapter discusses the management of audio, image, and video data stored in BLOBs only.

This chapter includes these sections:

- [Understanding the Performance Profile of Oracle Multimedia Operations](#)
- [Choosing LOB Storage Parameters for Multimedia LOBs](#)
- [Setting Database Initialization Parameters](#)

See Also:

Oracle Database SecureFiles and Large Objects Developer's Guide for more information about performance tuning when using LOBs in Oracle Database

6.1 Understanding the Performance Profile of Oracle Multimedia Operations

Multimedia data, and the operations that can be performed on that data, differs significantly from traditional types of data commonly stored in relational databases. A basic understanding of the performance profile of Oracle Multimedia operations can help you make better decisions when tuning your database for media performance.

The tables in this section summarize the general performance profiles for a set of commonly performed operations. There are two primary components to each profile. The I/O pattern is a general characterization of the primary type of I/O access and of how much of the media data the operation reads or writes. Because some operations involve two media objects, the I/O pattern is described for both the source and destination media objects. The second component is a general characterization of the level of CPU usage for the operation.



Note:

The information in these tables describes general characterizations and I/O patterns, thus CPU usage may vary considerably for some media formats.

The following table shows the profile for loading and retrieving data, which applies to all Oracle Multimedia media types.

Table 6-1 Performance Profile For All Multimedia Types

Operation	I/O Pattern (Source)	I/O Pattern (Destination)	I/O Pattern (Amount)	CPU Usage
Load new media data into a database	N/A	Sequential write	All	Low
Retrieve media from a database	Sequential read	N/A	All	Low

The following table shows the profile for commonly used functions and procedures of the ORD_IMAGE PL/SQL package.

Table 6-2 Performance Profile for ORD_IMAGE PL/SQL Package Functions and Procedures

Package Procedure	I/O Pattern (Source)	I/O Pattern (Destination)	I/O Pattern (Amount)	CPU Usage
getProperties()	Sequential read	N/A	Media header	Low to medium
getMetadata()	Sequential read	N/A	Media header	Low to medium
putMetadata()	Sequential read	Sequential write	All	Low to medium
process()	Sequential read	Sequential write	All	High
processCopy()	Sequential read	Sequential write	All	High
convert()	Sequential read	Sequential write	All	High
crop()	Sequential read	Sequential write	All	High
scale()	Sequential read	Sequential write	All	High
thumbnail()	Sequential read	Sequential write	All	High

The following table shows the profile for commonly used methods of the ORDImage type.

Table 6-3 Performance Profile For ORDImage Methods

Object Method	I/O Pattern (Source)	I/O Pattern (Destination)	I/O Pattern (Amount)	CPU Usage
setProperty()	Sequential read	N/A	Media header	Low to medium
getMetadata()	Sequential read	N/A	Media header	Low to medium
putMetadata()	Sequential read	Sequential write	All	Low to medium
process()	Sequential read	Sequential write	All	High
processCopy()	Sequential read	Sequential write	All	High

The following table shows the profile for commonly used functions and procedures of the ORD_DICOM PL/SQL package.

Table 6-4 Performance Profile for ORD_DICOM PL/SQL Package Functions and Procedures

Package Procedure	I/O Pattern (Source)	I/O Pattern (Destination)	I/O Pattern (Amount)	CPU Usage
extractMetadata()	Sequential read	N/A	Media header	Low to medium
writeMetadata()	Sequential read	Sequential write	All	Low to medium
makeAnonymous()	Sequential read	Sequential write	All	Low to medium
process()	Sequential read	Sequential write	All	High
processCopy()	Sequential read	Sequential write	All	High

The following table shows the profile for commonly used methods of the ORDDicom type.

Table 6-5 Performance Profile For ORDDicom Methods

Object Method	I/O Pattern (Source)	I/O Pattern (Destination)	I/O Pattern (Amount)	CPU Usage
setProperty()	Sequential read	N/A	Media header	Low to medium
extractMetadata()	Sequential read	N/A	Media header	Low to medium
writeMetadata()	Sequential read	Sequential write	All	Low to medium
makeAnonymous()	Sequential read	Sequential write	All	Low to medium
process()	Sequential read	Sequential write	All	High
processCopy()	Sequential read	Sequential write	All	High

The following table shows the profile for commonly used procedures of the ORD_AUDIO, ORD_DOC, and ORD_VIDEO PL/SQL packages.

Table 6-6 Performance Profile for ORD_AUDIO, ORD_DOC, and ORD_VIDEO PL/SQL Package Procedures

Package Procedure	I/O Pattern (Source)	I/O Pattern (Destination)	I/O Pattern (Amount)	CPU Usage
getProperties()	Sequential read	N/A	Media header	Low

The following table shows the profile for commonly used methods of the ORDAudio, ORDDoc, and ORDVideo types.

Table 6-7 Performance Profile For ORDAudio, ORDDoc, and ORDVideo Methods

Object Method	I/O Pattern (Source)	I/O Pattern (Destination)	I/O Pattern (Amount)	CPU Usage
setProperties()	Sequential read	N/A	Media header	Low
getProperties()	Sequential read	N/A	Media header	Low

6.2 Choosing LOB Storage Parameters for Multimedia LOBs

The choices you make for specifying LOB storage attributes during table creation can significantly affect the performance of media load, retrieval, and processing operations.

This section describes the most important options to consider and shows how the performance profile of Oracle Multimedia operations can affect the choice of LOB storage parameters.

The following subsections describe the LOB storage parameters and include examples of how to use them:

- [SecureFiles LOBs](#)
- [TABLESPACE](#)
- [CACHE, NOCACHE, and CACHE READS](#)
- [LOGGING and NOLOGGING](#)
- [Example of Setting LOB Storage Options](#)

See Also:

Oracle Database SecureFiles and Large Objects Developer's Guide for detailed information about LOBs

6.2.1 SecureFiles LOBs

Store your media data with SecureFiles LOBs.

Oracle recommends using SecureFiles LOBs for storing media data whenever possible. SecureFiles LOBs are identified by specifying the SQL parameter `SECUREFILE`.

6.2.2 TABLESPACE

You can achieve the best performance for LOBs by specifying storage for LOBs in a different tablespace than the one used for the table that contains the LOB.

If many different LOBs are to be accessed frequently, you can also specify a separate tablespace for each LOB column or attribute to reduce device contention.

6.2.3 CACHE, NOCACHE, and CACHE READS

The cache option is a part of the `STORE AS` clause, and determines whether LOB pages are stored in the buffer cache.

These are the possible values for the cache option:

- For the value `CACHE`, Oracle places LOB pages in the buffer cache where they can be shared among multiple users. Over time and if the LOB pages are no longer accessed, the pages are eventually removed from the buffer cache.
- For the value `NOCACHE`, LOB pages are not placed in the buffer cache.
- For the value `CACHE READS`, LOB pages are placed in the cache for read operations only.

If your application performs multiple read operations on a media object (for example: invoking the `getProperties()` procedure or the `setProperties()` method and then generating a thumbnail image), enable read caching for the source media LOB.

6.2.4 LOGGING and NOLOGGING

The logging option is a part of the `STORE AS` clause, and determines if REDO data is logged when a LOB is updated.

If the `[NO]LOGGING` clause is omitted, neither `NOLOGGING` nor `LOGGING` is specified and the logging attribute of the table or table partition defaults to the logging attribute of the tablespace in which it resides.

There is another alternative depending on how the cache option is specified.

- If `CACHE` is specified and `[NO]LOGGING` is omitted, `LOGGING` is automatically implemented (because you cannot have `CACHE NOLOGGING`).
- If `CACHE` is not specified and `[NO]LOGGING` is omitted, the `[NO]LOGGING` value is obtained from the tablespace in which the LOB segment resides.

Specify `NOLOGGING` only when you do not care about media recovery. However, if the disk, tape, or storage media fails, you will not be able to recover your changes from the log because those changes were not logged.

NOLOGGING can be useful for bulk loading of media data. For instance, when loading data into the LOB, if you do not care about the redo operation and you can start the load over if it fails, set the LOB data segment storage characteristics to NOCACHE NOLOGGING. This option provides good performance for the initial loading of data.

After you finish loading data, if necessary, you can use the ALTER TABLE statement to modify the LOB storage characteristics for the LOB data segment for normal LOB operations (for example: to CACHE or NOCACHE LOGGING).



Note:

Oracle Data Guard Redo Apply technology uses logging to populate the standby database. Thus, do not specify NOLOGGING when using this Data Guard technology.

6.2.5 Example of Setting LOB Storage Options

This section describes a simple example that shows how to use the performance profiles of Oracle Multimedia operations to guide your usage of LOB storage options.

In this example, Company X wants to build an archive for digital images. The archive stores a full resolution copy of the original image, and two Web-ready, JPEG format versions of the original at reduced scales, one at 50% of the original size and another at 25% of the original size. The database team plans to use the SQL*Loader utility to bulk load all the initial images. Then, they can use a PL/SQL program to initialize the image data. Initialization consists of setting the properties for the original image and generating the scaled images. After initialization, the table is prepared for the primary application, which retrieves images for Web-based users.

The following example shows a table definition for storing the images. The table stores the binary image data using SecureFiles in tablespace `tbs2`. All the other table data is stored in tablespace `tbs1`. Although this example uses the `ORDImage` object type for storing the images, the concepts apply to images stored directly in BLOB columns.

```
create table images(id          integer primary key,
                  original    ordsys.ordimage,
                  scale50     ordsys.ordimage,
                  scale25     ordsys.ordimage)
tablespace tbs1
lob(original.source.localdata)store as secureFile (tablespace tbs2)
lob(scale50.source.localdata)store as secureFile (tablespace tbs2)
lob(scale25.source.localdata)store as secureFile (tablespace tbs2);
```

After the table is created, the image data can be loaded. Loading image data generates a sequential write pattern to the LOB. Because no applications are reading the data during the load operation, caching it is not required. You can also improve load performance by disabling logging for the column that is loaded. The following command dynamically alters the table to prepare the original image column LOB for loading.

```
alter table images modify lob(original.source.localdata) (nocache nologging);
```

After loading, the next step is to set the image properties for the `original` column and generate the scaled versions to be stored in the `scale50` and `scale25` columns. In this step, the source images are fully read twice to generate the scaled versions. The

scaled images that are generated are written but not read. The following command dynamically alters the table to enable read caching for the source image, and disables caching and logging for the destination images.

```
alter table images modify lob(original.source.localdata) (cache reads);
alter table images modify lob(scale50.source.localdata) (nocache nologging);
alter table images modify lob(scale25.source.localdata) (nocache nologging);
```

After running the program to set the properties of the `original` image and generate the scaled versions, the LOB storage attributes can be optimized for the main application that retrieves images for users to view in a Web browser. Because the archive contains millions of images, users are not expected to view the same image simultaneously. Thus, there is little benefit to caching the image data. The following command reenables logging for the LOBs and disables caching.

```
alter table images modify lob(original.source.localdata) (nocache logging);
alter table images modify lob(scale50.source.localdata) (nocache logging);
alter table images modify lob(scale25.source.localdata) (nocache logging);
```

 **See Also:**

[Table 6-1](#) through [Table 6-7](#) for more information about Oracle Multimedia performance profiles

6.3 Setting Database Initialization Parameters

You can disable logging of LOB data at the column level to reduce the amount of I/O to the redo log. If logging cannot be disabled, additional database tuning may be required. For example, you may have to increase the size of the redo log buffer to prevent load processes from waiting.

The initialization parameter `LOG_BUFFER` specifies the amount of memory (in bytes) that Oracle uses when buffering redo entries to a redo log file. Redo log entries contain a record of the changes that have been made to the database block buffers. The LGWR process writes redo log entries from the log buffer to a redo log file.

 **See Also:**

- *Oracle Database Performance Tuning Guide* for more information about configuring the database redo log buffer
- *Oracle Database Reference* for comprehensive information about setting database initialization parameters
- *Oracle Database Administrator's Guide* for more information about managing initialization parameters

A

Managing Oracle Multimedia Installations

This appendix describes how to manage Oracle Multimedia installations.

This appendix includes these sections:

- [Oracle Multimedia Installed Users and Privileges](#)
- [Installing and Configuring Oracle Multimedia](#)
- [Verifying an Installed Version of Oracle Multimedia](#)
- [Upgrading an Installed Version of Oracle Multimedia](#)
- [Downgrading an Installed Version of Oracle Multimedia](#)

Note:

See the Oracle Multimedia `README.txt` file located in `<ORACLE_HOME>/ord/im/admin` for the latest information

A.1 Oracle Multimedia Installed Users and Privileges

This topic summarizes the database user accounts created by the Oracle Multimedia installation procedure.

The Oracle Multimedia installation procedure performs these functions:

- Creates the database users shown in [Table A-1](#) with the privileges required by Oracle Multimedia.

Table A-1 Installed Database Users

Name of User	Type of User
ORDSYS	Oracle Multimedia
ORDPLUGINS	Oracle Multimedia
SI_INFORMTN_SCHEMA	Oracle Multimedia
ORDDATA	Oracle Multimedia
MDSYS	Oracle Spatial and Graph/Oracle Locator

- Creates the default passwords shown in [Table A-2](#) for the Oracle Multimedia and MDSYS user accounts, and then locks the accounts and marks their default passwords as expired.

Table A-2 User Accounts and Default Passwords

User Account	Installation Password
ORDSYS	<i>ORDSYS</i>
ORDPLUGINS	<i>ORDPLUGINS</i>
SI_INFORMTN_SCHEMA	<i>SI_INFORMTN_SCHEMA</i>
ORDDATA	<i>ORDDATA</i>
MDSYS	<i>MDSYS</i>

▲ Caution:

Oracle does not recommend logging in directly to the user accounts shown in [Table A-2](#).

- Grants the EXECUTE privilege to the user group PUBLIC for the Oracle Multimedia packages and objects installed in these schemas:
 - ORDSYS
 - ORDPLUGINS
 - SI_INFORMTN_SCHEMA
 - MDSYS

A.2 Installing and Configuring Oracle Multimedia

Oracle Multimedia is automatically installed and configured with Oracle Database.

The following subsections describe the steps to perform before manual installation and configuration of Oracle Multimedia, and the steps for manually installing and configuring it:

- [Preinstallation Steps](#)
- [Installation and Configuration Steps](#)

 **Caution:**

Performing any of these unsupported and prohibited actions could cause internal errors and security violations in the database management system.

These users are created during database installation, and might change in future releases:

- Users in which Oracle-supplied Oracle Multimedia is installed: `ORDSYS`, `ORDPLUGINS`, `SI_INFORMTN_SCHEMA`, and `ORDDATA`
- User in which Oracle Multimedia Locator is installed if Oracle Spatial and Graph is not installed: `MDSYS`

Do not delete any of these users.

Do not connect to, modify, or change the privileges of any of these users or their contents (which are supplied by Oracle Multimedia and reserved by Oracle), with these exceptions:

- You can add user-defined packages to the user `ORDPLUGINS`.
- DICOM administrators can store user-defined DICOM data model configuration documents in the user `ORDDATA`, using the DICOM data model repository API.

Related Topics

- [Extending Oracle Multimedia](#)
This appendix describes various methods for extending the Oracle Multimedia object types.

 **See Also:**

Oracle Multimedia DICOM Developer's Guide for more information about inserting documents into the data model repository

A.2.1 Preinstallation Steps

Installing Oracle Multimedia manually requires a few preliminary tasks.

Before installing and configuring Oracle Multimedia manually, perform these steps:

1. Install Oracle Database, including PL/SQL, Oracle JVM, Oracle XML Database, and Oracle XDK.
2. Create the database.
3. Start the database.
4. Verify that the required software is correctly installed and valid, as follows:
 - a. Run SQL*Plus, connect as SYSDBA, and enter these queries:


```
SQL> select version, status from dba_registry where comp_id='JAVAVM';
SQL> select version, status from dba_registry where comp_id='XDB';
SQL> select version, status from dba_registry where comp_id='XML';
```

- b. Examine the results of the queries to ensure that each `version` value is identical to the version of Oracle Multimedia that you are installing and each `status` value is `VALID`.

 **See Also:**

Oracle Database Installation Guide for your operating system for more detailed information

A.2.2 Installation and Configuration Steps

Installing and configuring Oracle Multimedia manually requires a few setup tasks.

These steps are not required if you use the Database Configuration Assistant.

To install and configure Oracle Multimedia manually, perform these steps:

 **Note:**

In these steps, `<ORACLE_HOME>` is replaced with the location of the Oracle home directory.

1. Use Oracle Universal Installer to install the files that comprise Oracle Multimedia on your system.
2. Decide which tablespace to use for the Oracle Multimedia users, and which tablespace to use for the Oracle Spatial and Graph/Oracle Multimedia Location Services user (see [Table A-1](#)). Oracle recommends using the `SYSAUX` tablespace for all these users.
3. Use the Perl script `catcon.pl` to run the Oracle Multimedia installation scripts. The `catcon.pl` script is in the directory `<ORACLE_HOME>/rdtms/admin`.

- a. Run the script to create the users and grant the appropriate privileges.

If you are using a tablespace other than `SYSAUX`, replace the `SYSAUX` parameters with the tablespaces you have chosen. The first parameter is the tablespace for Oracle Multimedia; the second parameter is the tablespace for Oracle Spatial and Graph.

```
perl catcon.pl -u SYS -d <ORACLE_HOME>/ord/admin -b ordinst ordinst.sql '--pSYSAUX' '--pSYSAUX'
```

- b. Run the script to install the Oracle Multimedia types and packages.

```
perl catcon.pl -u SYS -d <ORACLE_HOME>/ord/im/admin -b catim catim.sql
```

Now Oracle Multimedia is ready for use.

A.3 Verifying an Installed Version of Oracle Multimedia

After installing or upgrading Oracle Multimedia, you can verify the Oracle Multimedia installation by calling the Oracle Multimedia validation procedure.

To run the Oracle Multimedia validation procedure, perform these steps:

1. Start SQL*Plus and connect as SYSDBA.
2. Execute the procedure `sys.validate_ordim`:

```
SQL> execute sys.validate_ordim;
```

If the validation procedure detects invalid objects, it lists the first few invalid objects and sets the registry entry to `INVALID`; otherwise, it silently sets the Oracle Multimedia registry entry to `VALID`.

3. Verify that the registry entry for Oracle Multimedia is correct, as follows:
 - a. Enter this query from SQL*Plus, while you are connected as SYSDBA:

```
SQL> select version, status from dba_registry where comp_id='ORDIM';
```

- b. Examine the result of the query to ensure that the `version` value is correct and the `status` value is `VALID`.

A.4 Upgrading an Installed Version of Oracle Multimedia

Oracle Multimedia is automatically upgraded when you upgrade a database with the Oracle Multimedia feature installed.

See Also:

Oracle Database Upgrade Guide for detailed upgrading instructions

A.5 Downgrading an Installed Version of Oracle Multimedia

Oracle Multimedia is automatically downgraded when you downgrade a database with the Oracle Multimedia feature installed.

Caution:

Do not modify your DICOM data model repository until you are sure that you are not going to downgrade from the new release of Oracle Database back to the source release.

Changes to the Oracle Multimedia DICOM data model repository (such as document insertions or deletions) that you make after a database upgrade are lost after a database downgrade.

 **See Also:**

Oracle Database Upgrade Guide for detailed downgrading instructions

B

Extending Oracle Multimedia

This appendix describes various methods for extending the Oracle Multimedia object types.

Note:

The ability to extend Oracle Multimedia object types to support new source types and new format types is deprecated in the current release, and may be desupported in a future release. Oracle recommends that you stop using deprecated features as soon as possible.

Oracle Multimedia object types can be extended to support:

- Other external sources of media data not currently supported
- Other media data formats not currently supported
- Media (audio and video) data processing

For each unique external media data source or each unique ORDAudio, ORDDoc, or ORDVideo data format you want to support, you must:

1. Design your new data source or new ORDAudio, ORDDoc, or ORDVideo data format.
2. Implement your new data source or new ORDAudio, ORDDoc, or ORDVideo data format.
3. Install your new source package into the `ORDPLUGINS` schema.
4. Grant EXECUTE privileges on your new source package to `PUBLIC`.

This appendix includes these sections:

- [Supporting Other External Sources](#)
- [Supporting Other Media Data Formats](#)
- [Supporting Media Data Processing](#)

B.1 Supporting Other External Sources

You can extend the Oracle Multimedia object types to support new external data sources by implementing a new source package.

To implement your new data source, you must implement the required interfaces in the `ORDX_<srcType>_SOURCE` package in the `ORDPLUGINS` schema (where `<srcType>` represents the name of the new external source type). Use the package body example described in "Extending Oracle Multimedia to Support a New Data Source" as a template to create the package body. Then, set the source type parameter in the `setSourceInformation()` call to the appropriate source value to indicate to the

ORDAudio, ORDImage, ORDDoc, or ORDVideo object that package `ORDPLUGINS.ORDX_<srcType>_SOURCE` should be used. Use the `ORDPLUGINS.ORDX_FILE_SOURCE` and `ORDPLUGINS.ORDX_HTTP_SOURCE` packages as guides when you extend support to other external audio, image, video, or other heterogeneous media data sources.

The following subsection presents reference information about predefined and user-defined packages:

- [External Source Packages](#)

Related Topics

- [Extending Oracle Multimedia to Support a New Data Source](#)
Extending Oracle Multimedia to support a new data source requires a series of tasks.

B.1.1 External Source Packages

Name your PL/SQL package as `ORDX_<name>_<module_name>`, where the `<module_name>` is `SOURCE` for `ORDSource`. Use the `ORDPLUGINS.ORDX_FILE_SOURCE` and `ORDPLUGINS.ORDX_HTTP_SOURCE` packages as a guide in developing your new source type package.

For example, the FILE source type package described in "ORDPLUGINS.ORDX_FILE_SOURCE Package" is named `ORDX_FILE_SOURCE` and the HTTP source type package described in "ORDPLUGINS.ORDX_HTTP_SOURCE Package" is named `ORDX_HTTP_SOURCE`, and `<name>` is the source type. Both source type names, FILE and HTTP, are reserved to Oracle.

The following subsections provide examples and more information about extending the supported external sources of audio, image, video, or other heterogeneous media data:

- [ORDPLUGINS.ORDX_FILE_SOURCE Package](#)
- [ORDPLUGINS.ORDX_HTTP_SOURCE Package](#)
- [Extending Oracle Multimedia to Support a New Data Source](#)

B.1.1.1 ORDPLUGINS.ORDX_FILE_SOURCE Package

The `ORDPLUGINS.ORDX_FILE_SOURCE` package provides support for multimedia stored in the local file system external to the database.

The `ORDPLUGINS.ORDX_FILE_SOURCE` package is defined as follows:

```
CREATE OR REPLACE PACKAGE ORDX_FILE_SOURCE AS
  -- functions/procedures
  FUNCTION processCommand(obj      IN OUT NOCOPY ORDSYS.ORDSource,
                        ctx      IN OUT RAW,
                        cmd      IN VARCHAR2,
                        arglist  IN VARCHAR2,
                        result   OUT RAW)
    RETURN RAW;
  PROCEDURE import(obj      IN OUT NOCOPY ORDSYS.ORDSource,
                  ctx      IN OUT RAW,
                  mimetype  OUT VARCHAR2,
                  format    OUT VARCHAR2);
```

```

PROCEDURE import(obj      IN OUT NOCOPY ORDSYS.ORDSource,
                 ctx      IN OUT RAW,
                 dlob     IN OUT NOCOPY BLOB,
                 mimetype OUT VARCHAR2,
                 format   OUT VARCHAR2);
PROCEDURE importFrom(obj  IN OUT NOCOPY ORDSYS.ORDSource,
                     ctx   IN OUT RAW,
                     mimetype OUT VARCHAR2,
                     format  OUT VARCHAR2,
                     loc    IN VARCHAR2,
                     name   IN VARCHAR2);
PROCEDURE importFrom(obj  IN OUT NOCOPY ORDSYS.ORDSource,
                     ctx   IN OUT RAW,
                     dlob  IN OUT NOCOPY BLOB,
                     mimetype OUT VARCHAR2,
                     format  OUT VARCHAR2,
                     loc    IN VARCHAR2,
                     name   IN VARCHAR2);
PROCEDURE export(obj  IN OUT NOCOPY ORDSYS.ORDSource,
                 ctx  IN OUT RAW,
                 slob IN OUT NOCOPY BLOB,
                 loc  IN VARCHAR2,
                 name IN VARCHAR2);
FUNCTION getContenLength(obj  IN ORDSYS.ORDSource,
                        ctx  IN OUT RAW),
        RETURN INTEGER;
PRAGMA RESTRICT_REFERENCES(getContenLength, WNDS, WNPS, RNDS, RNPS);
FUNCTION getSourceAddress(obj  IN ORDSYS.ORDSource,
                        ctx  IN OUT RAW,
                        userData IN VARCHAR2)
        RETURN VARCHAR2;
PRAGMA RESTRICT_REFERENCES(getSourceAddress, WNDS, WNPS, RNDS, RNPS);

FUNCTION open(obj  IN OUT NOCOPY ORDSYS.ORDSource,
             userArg IN RAW,
             ctx  OUT RAW) RETURN INTEGER;
FUNCTION close(obj  IN OUT NOCOPY ORDSYS.ORDSource, ctx  IN OUT RAW)
        RETURN INTEGER;
FUNCTION trim(obj  IN OUT NOCOPY ORDSYS.ORDSource,
             ctx  IN OUT RAW,
             newlen IN INTEGER) RETURN INTEGER;
PROCEDURE read(obj  IN OUT NOCOPY ORDSYS.ORDSource,
              ctx  IN OUT RAW,
              startPos IN INTEGER,
              numBytes IN OUT INTEGER,
              buffer  OUT RAW);
PROCEDURE write(obj  IN OUT NOCOPY ORDSYS.ORDSource,
               ctx  IN OUT RAW,
               startPos IN INTEGER,
               numBytes IN OUT INTEGER,
               buffer  OUT RAW);
END ORDX_FILE_SOURCE;
/

```

Table B-1 shows the methods that are supported in the `ORDX_FILE_SOURCE` package and the exception that is raised if you call a method that is not supported.

Table B-1 Methods Supported in the ORDPLUGINS.ORDX_FILE_SOURCE Package

Name of Method	Level of Support
processCommand	Not supported - raises exception: METHOD_NOT_SUPPORTED
import	Supported
import	Supported
importFrom	Supported
importFrom	Supported
export	Supported
getContentLength	Supported
getSourceAddress	Supported
open	Supported
close	Supported
trim	Not supported - raises exception: METHOD_NOT_SUPPORTED
read	Supported
write	Not supported - raises exception: METHOD_NOT_SUPPORTED

B.1.1.2 ORDPLUGINS.ORDX_HTTP_SOURCE Package

The `ORDPLUGINS.ORDX_HTTP_SOURCE` package provides support for multimedia stored in any HTTP server and accessed through a URL.

The `ORDPLUGINS.ORDX_HTTP_SOURCE` package is defined as follows:

```
CREATE OR REPLACE PACKAGE ORDX_HTTP_SOURCE AS
-- functions/procedures
FUNCTION processCommand(obj      IN OUT NOCOPY ORDSYS.ORDSource,
                        ctx      IN OUT RAW,
                        cmd      IN VARCHAR2,
                        arglist  IN VARCHAR2,
                        result   OUT RAW)
    RETURN RAW;
PROCEDURE import(obj      IN OUT NOCOPY ORDSYS.ORDSource,
                 ctx      IN OUT RAW,
                 mimetype  OUT VARCHAR2,
                 format    OUT VARCHAR2);
PROCEDURE import(obj      IN OUT NOCOPY ORDSYS.ORDSource,
                 ctx      IN OUT RAW,
                 dlob     IN OUT NOCOPY BLOB,
                 mimetype  OUT VARCHAR2,
                 format    OUT VARCHAR2);
PROCEDURE importFrom(obj      IN OUT NOCOPY ORDSYS.ORDSource,
                    ctx      IN OUT RAW,
                    mimetype  OUT VARCHAR2,
                    format    OUT VARCHAR2,
                    loc      IN VARCHAR2,
                    name     IN VARCHAR2);
PROCEDURE importFrom(obj      IN OUT NOCOPY ORDSYS.ORDSource,
                    ctx      IN OUT RAW,
                    dlob     IN OUT NOCOPY BLOB,
                    mimetype  OUT VARCHAR2,
```

```

        format    OUT VARCHAR2,
        loc       IN VARCHAR2,
        name     IN VARCHAR2);
PROCEDURE export(obj IN OUT NOCOPY ORDSYS.ORDSource,
                ctx IN OUT RAW,
                dlob IN OUT NOCOPY BLOB,
                loc IN VARCHAR2,
                name IN VARCHAR2);
FUNCTION getContentLength(obj IN ORDSYS.ORDSource,
                        ctx IN OUT RAW)
    RETURN INTEGER;
PRAGMA RESTRICT_REFERENCES(getContentLength, WNDS, WNPS, RNDS, RNPS, TRUST);
FUNCTION getSourceAddress(obj IN ORDSYS.ORDSource,
                        ctx IN OUT RAW,
                        userData IN VARCHAR2)
    RETURN VARCHAR2;
PRAGMA RESTRICT_REFERENCES(getSourceAddress, WNDS, WNPS, RNDS, RNPS);
FUNCTION open(obj IN OUT NOCOPY ORDSYS.ORDSource, userArg IN RAW,
             ctx OUT RAW) RETURN INTEGER;
FUNCTION close(obj IN OUT NOCOPY ORDSYS.ORDSource, ctx IN OUT RAW)
    RETURN INTEGER;
FUNCTION trim(obj IN OUT NOCOPY ORDSYS.ORDSource,
             ctx IN OUT RAW,
             newlen IN INTEGER) RETURN INTEGER;
PROCEDURE read(obj IN OUT NOCOPY ORDSYS.ORDSource,
              ctx IN OUT RAW,
              startPos IN INTEGER,
              numBytes IN OUT INTEGER,
              buffer OUT RAW);
PROCEDURE write(obj IN OUT NOCOPY ORDSYS.ORDSource,
               ctx IN OUT RAW,
               startPos IN INTEGER,
               numBytes IN OUT INTEGER,
               buffer OUT RAW);
END ORDX_HTTP_SOURCE;
/

```

Table B-2 shows the methods that are supported in the `ORDX_HTTP_SOURCE` package and the exception that is raised if you call a method that is not supported.

Table B-2 Methods Supported in the `ORDPLUGINS.ORDX_HTTP_SOURCE` Package

Name of Method	Level of Support
<code>processCommand</code>	Not supported - raises exception: <code>METHOD_NOT_SUPPORTED</code>
<code>import</code>	Supported
<code>import</code>	Supported
<code>importFrom</code>	Supported
<code>importFrom</code>	Supported
<code>export</code>	Not supported - raises exception: <code>METHOD_NOT_SUPPORTED</code>
<code>getContentLength</code>	Supported
<code>getSourceAddress</code>	Supported
<code>open</code>	Supported
<code>close</code>	Supported

**Table B-2 (Cont.) Methods Supported in the
ORDPLUGINS.ORDX_HTTP_SOURCE Package**

Name of Method	Level of Support
trim	Not supported - raises exception: METHOD_NOT_SUPPORTED
read	Not supported - raises exception: METHOD_NOT_SUPPORTED
write	Not supported - raises exception: METHOD_NOT_SUPPORTED

B.1.1.3 Extending Oracle Multimedia to Support a New Data Source

Extending Oracle Multimedia to support a new data source requires a series of tasks.

Follow these steps to extend Oracle Multimedia to support a new data source:

1. Design your new data source.
2. Implement your new data source and name it, for example, `ORDX_MY_SOURCE.SQL`.
3. Install your new `ORDX_MY_SOURCE.SQL` package into the `ORDPLUGINS` schema.
4. Grant `EXECUTE` privileges on your new package, for example, `ORDX_MY_SOURCE.SQL`, to `PUBLIC`.
5. Set the `src`type to `my` to cause your package to be invoked.

Example B-1 Package Body for Extending Support to a New Data Source

```
CREATE OR REPLACE PACKAGE BODY ORDX_MY_SOURCE
AS
  -- functions/procedures
  FUNCTION processCommand(
    obj IN OUT NOCOPY ORDSYS.ORDSource,
    ctx IN OUT RAW,
    cmd IN VARCHAR2,
    arglist IN VARCHAR2,
    result OUT RAW)
    RETURN RAW
  IS
    --Your variables go here.
  BEGIN
    --Your code goes here.
  END processCommand;
  PROCEDURE import( obj IN OUT NOCOPY ORDSYS.ORDSource,
    ctx IN OUT RAW,
    mimetype OUT VARCHAR2,
    format OUT VARCHAR2)
  IS
    --Your variables go here.
  BEGIN
    --Your code goes here.
  END import;
  PROCEDURE import( obj IN OUT NOCOPY ORDSYS.ORDSource,
    ctx IN OUT RAW,
    dlob IN OUT NOCOPY BLOB,
    mimetype OUT VARCHAR2,
    format OUT VARCHAR2)
  IS
    --Your variables go here.
  BEGIN
```

```

--Your code goes here.
END import;
PROCEDURE importFrom( obj      IN OUT NOCOPY ORDSYS.ORDSource,
                     ctx      IN OUT RAW,
                     mimetype  OUT VARCHAR2,
                     format   OUT VARCHAR2,
                     loc      IN VARCHAR2,
                     name     IN VARCHAR2)

IS
--Your variables go here.
BEGIN
--Your code goes here.
END importFrom;
PROCEDURE importFrom( obj      IN OUT NOCOPY ORDSYS.ORDSource,
                     ctx      IN OUT RAW,
                     dlob     IN OUT NOCOPY BLOB,
                     mimetype  OUT VARCHAR2,
                     format   OUT VARCHAR2,
                     loc      IN VARCHAR2,
                     name     IN VARCHAR2)

IS
--Your variables go here.
BEGIN
--Your code goes here.
END importFrom;
PROCEDURE export( obj IN OUT NOCOPY ORDSYS.ORDSource,
                 ctx IN OUT RAW,
                 slob IN OUT NOCOPY BLOB,
                 loc IN VARCHAR2,
                 name IN VARCHAR2)

IS
--Your variables go here.
BEGIN
--Your code goes here.
END export;

FUNCTION getContentLength( obj IN ORDSYS.ORDSource,
                          ctx IN OUT RAW)

RETURN INTEGER
IS
--Your variables go here.
BEGIN
--Your code goes here.
END getContentLength;
FUNCTION getSourceAddress(obj IN ORDSYS.ORDSource,
                        ctx IN OUT RAW,
                        userData IN VARCHAR2)

RETURN VARCHAR2
IS
--Your variables go here.
BEGIN
--Your code goes here.
END getSourceAddress;
FUNCTION open(obj IN OUT NOCOPY ORDSYS.ORDSource, userArg IN RAW, ctx OUT RAW)
RETURN INTEGER
IS
--Your variables go here.
BEGIN
--Your code goes here.
END open;
FUNCTION close(obj IN OUT NOCOPY ORDSYS.ORDSource, ctx IN OUT RAW)

```

```

RETURN INTEGER
IS
--Your variables go here.
BEGIN
--Your code goes here.
END close;
FUNCTION trim(obj      IN OUT NOCOPY ORDSYS.ORDSource,
              ctx      IN OUT RAW,
              newlen   IN INTEGER)

RETURN INTEGER
IS
--Your variables go here.
BEGIN
--Your code goes here.
END trim;
PROCEDURE read(obj      IN OUT NOCOPY ORDSYS.ORDSource,
               ctx      IN OUT RAW,
               startPos IN INTEGER,
               numBytes IN OUT INTEGER,
               buffer   OUT RAW)

IS
--Your variables go here.
BEGIN
--Your code goes here.
END read;
PROCEDURE write(obj      IN OUT NOCOPY ORDSYS.ORDSource,
                ctx      IN OUT RAW,
                startPos IN INTEGER,
                numBytes IN OUT INTEGER,
                buffer   OUT RAW)

IS
--Your variables go here.
BEGIN
--Your code goes here.
END write;
END ORDX_MY_SOURCE;
/
show errors;

```

A package body listing is provided in [Example B-1](#) to assist you in this operation. Add your variables to the places that say "--Your variables go here" and add your code to the places that say "--Your code goes here".

B.2 Supporting Other Media Data Formats

You can extend support for the ORDAudio, ORDDoc, and ORDVideo object types to include new media data formats by defining new format packages.

To implement your new ORDAudio, ORDDoc, or ORDVideo data format, you must implement the required interfaces in the `ORDPLUGINS.ORDX_<format>_<media>` package in the `ORDPLUGINS` schema (where `<format>` represents the name of the new audio or video, or other heterogeneous media data format and `<media>` represents the type of media ("AUDIO" or "VIDEO", or "DOC"). Use the `ORDPLUGINS.ORDX_DEFAULT_<media>` package as a guide when you extend support to other audio or video data formats or other heterogeneous media data formats. Use the package body examples described in "Extending Oracle Multimedia to Support a New Audio Data Format", "Extending Oracle Multimedia to Support a New ORDDoc Data Format", and "Extending Oracle Multimedia to Support a New Video Data Format" as templates to create the audio or

video, or other heterogeneous media data package body, respectively. Then, set the new format parameter in the `setFormat()` call to the appropriate format value to indicate to the `ORDAudio`, `ORDDoc`, or `ORDVideo` object that the package `ORDPLUGINS.ORDX_<format>_<media>` must be used for method invocation.

The following subsections describe how to extend Oracle Multimedia to support other data formats:

- [Supporting Other ORDAudio Data Formats](#)
- [Supporting Other ORDDoc Data Formats](#)
- [Supporting Other Video Data Formats](#)
- [Supporting Other Image Data Formats](#)

B.2.1 Supporting Other ORDAudio Data Formats

The following subsections describe how to extend `ORDAudio` to support other data formats:

- [ORDPLUGINS.ORDX_DEFAULT_AUDIO Package](#)
- [Extending Oracle Multimedia to Support a New Audio Data Format](#)

B.2.1.1 ORDPLUGINS.ORDX_DEFAULT_AUDIO Package

Use the `ORDPLUGINS.ORDX_DEFAULT_AUDIO` package as a guide in developing your own `ORDPLUGINS.ORDX_<format>_AUDIO` audio format package. This package sets the `mimeType` field in the `setProperties()` method with a MIME type value that is dependent on the file format.

The `ORDPLUGINS.ORDX_DEFAULT_AUDIO` package is defined as follows:

```
CREATE OR REPLACE PACKAGE ORDX_DEFAULT_AUDIO
authid current_user
AS
--AUDIO ATTRIBUTES ACCESSORS

PROCEDURE setProperties(ctx IN OUT RAW,
                      obj IN OUT NOCOPY ORDSYS.ORDAudio,
                      setComments IN NUMBER := 0);
FUNCTION checkProperties(ctx IN OUT RAW, obj IN OUT ORDSYS.ORDAudio)
RETURN NUMBER;
FUNCTION getAttribute(ctx IN OUT RAW,
                    obj IN ORDSYS.ORDAudio,
                    name IN VARCHAR2) RETURN VARCHAR2;
PROCEDURE getAllAttributes(ctx IN OUT RAW,
                          obj IN ORDSYS.ORDAudio,
                          attributes IN OUT NOCOPY CLOB);

--AUDIO PROCESSING METHODS
FUNCTION processCommand(ctx          IN OUT RAW,
                      obj           IN OUT NOCOPY ORDSYS.ORDAudio,
                      cmd           IN VARCHAR2,
                      arguments     IN VARCHAR2,
                      result        OUT RAW)

RETURN RAW;

END;
/
```

Table B-3 shows the methods that are supported in the `ORDPLUGINS.ORDX_DEFAULT_AUDIO` package and the exceptions that are raised if you call a method that is not supported.

Table B-3 Methods Supported in the `ORDPLUGINS.ORDX_DEFAULT_AUDIO` Package

Name of Method	Level of Support
<code>setProperty</code> s	Supported; if the source is local, extract attributes from the local data and set the properties, but if the source is <code>NULL</code> , raise an <code>ORDSYS.ORDSourceExceptions.EMPTY_SOURCE</code> exception; if the source is a <code>BFILE</code> , then extract attributes from the <code>BFILE</code> and set the properties; if the source is neither local nor a <code>BFILE</code> , get the media content into a temporary LOB, extract attributes from the data, and set the properties.
<code>checkProperties</code>	Supported; if the source is local, extract the attributes from the local data and compare them with the attribute values of the object, but if the source is <code>NULL</code> , raise an <code>ORDSYS.ORDSourceExceptions.EMPTY_SOURCE</code> exception; if the source is a <code>BFILE</code> , extract the attributes from the <code>BFILE</code> and compare them with the attribute values of the object; if the source is neither local nor a <code>BFILE</code> , get the media content into a temporary LOB, extract the attributes from the media content and compare them with the attribute values of the object.
<code>getAttribute</code>	Not supported - raises exceptions: <code>METHOD_NOT_SUPPORTED</code> and <code>AUDIO_PLUGIN_EXCEPTION</code> .
<code>getAllAttributes</code>	Supported; if the source is local, get the attributes and return them, but if the source is <code>NULL</code> , raise an <code>ORDSYS.ORDSourceExceptions.EMPTY_SOURCE</code> exception; otherwise, if the source is external, raise an <code>ORDSYS.ORDAudioExceptions.LOCAL_DATA_SOURCE_REQUIRED</code> exception.
<code>processCommand</code>	Not supported - raises exceptions: <code>METHOD_NOT_SUPPORTED</code> and <code>AUDIO_PLUGIN_EXCEPTION</code> .

B.2.1.2 Extending Oracle Multimedia to Support a New Audio Data Format

Extending Oracle Multimedia to support a new audio data format requires a series of tasks.

Follow these steps to extend Oracle Multimedia to support a new audio data format:

1. Design your new audio data format.
 - a. To support a new audio data format, implement the required interfaces in the `ORDX_<format>_AUDIO` package in the `ORDPLUGINS` schema (where `<format>` represents the name of the new audio data format). Use the package body example in [Example B-2](#) as a template to create the audio package body.
 - b. Then, set the new format parameter in the `setFormat()` call to the appropriate format value to indicate to the audio object that package `ORDPLUGINS.ORDX_<format>_AUDIO` should be called.
2. Implement your new audio data format and name it, for example, `ORDX_MY_AUDIO.SQL`.
3. Install your new `ORDX_MY_AUDIO.SQL` package into the `ORDPLUGINS` schema.

4. Grant EXECUTE privileges on your new package, for example, ORDX_MY_AUDIO, to PUBLIC.
5. In an application, set the format attribute to my to cause your package to be invoked.

A package body listing is provided in [Example B-2](#) to assist you in this operation. Add your variables to the places that say "--Your variables go here" and add your code to the places that say "--Your code goes here".

Example B-2 Package Body for Extending Support to a New Audio Data Format

```

CREATE OR REPLACE PACKAGE BODY ORDX_MY_AUDIO
AS
  --AUDIO ATTRIBUTES ACCESSORS
  PROCEDURE setProperties(ctx IN OUT RAW,
                        obj IN OUT NOCOPY ORDSYS.ORDAudio,
                        setComments IN NUMBER :=0)
  IS
  --Your variables go here.
  BEGIN
  --Your code goes here.
  END;
  FUNCTION checkProperties(ctx IN OUT RAW, obj IN OUT ORDSYS.ORDAudio)
  RETURN NUMBER
  IS
  --Your variables go here.
  BEGIN
  --Your code goes here.
  END;
  FUNCTION getAttribute(ctx IN OUT RAW,
                      obj IN ORDSYS.ORDAudio,
                      name IN VARCHAR2)
  RETURN VARCHAR2
  IS
  --Your variables go here.
  BEGIN
  --Your code goes here.
  END;
  PROCEDURE getAllAttributes(ctx IN OUT RAW,
                          obj IN ORDSYS.ORDAudio,
                          attributes IN OUT NOCOPY CLOB)
  IS
  --Your variables go here.
  BEGIN
  --Your code goes here.
  END;
  -- AUDIO PROCESSING METHODS
  FUNCTION processCommand(
                                ctx          IN OUT RAW,
                                obj          IN OUT NOCOPY ORDSYS.ORDAudio,
                                cmd          IN VARCHAR2,
                                arguments IN VARCHAR2,
                                result      OUT RAW)
  RETURN RAW
  IS
  --Your variables go here.
  BEGIN
  --Your code goes here.
  END;
END;

```

```
/
show errors;
```

Related Topics

- [ORDPLUGINS.ORDX_DEFAULT_AUDIO Package](#)
Use the `ORDPLUGINS.ORDX_DEFAULT_AUDIO` package as a guide in developing your own `ORDPLUGINS.ORDX_<format>_AUDIO` audio format package. This package sets the `mimeType` field in the `setProperties()` method with a MIME type value that is dependent on the file format.

B.2.2 Supporting Other ORDDoc Data Formats

The following subsections describe how to extend ORDDoc to support other data formats:

- [ORDPLUGINS.ORDX_DEFAULT_DOC Package](#)
- [Extending Oracle Multimedia to Support a New ORDDoc Data Format](#)

B.2.2.1 ORDPLUGINS.ORDX_DEFAULT_DOC Package

Use the `ORDPLUGINS.ORDX_DEFAULT_DOC` package as a guide in developing your own `ORDPLUGINS.ORDX_<format>_DOC` media format package.

The `ORDPLUGINS.ORDX_DEFAULT_DOC` package is defined as follows:

```
CREATE OR REPLACE PACKAGE ORDX_DEFAULT_DOC
authid current_user
AS

PROCEDURE setProperties(ctx IN OUT RAW,
                      obj IN OUT NOCOPY ORDSYS.ORDDoc,
                      setComments IN NUMBER := 0);

END;
```

[Table B-4](#) shows the method that is supported in the `ORDPLUGINS.ORDX_DEFAULT_DOC` package and the exception that is raised if the source is `NULL`.

Table B-4 Method Supported in the `ORDPLUGINS.ORDX_DEFAULT_DOC` Package

Name of Method	Level of Support
<code>setProperties</code>	Supported; if the source is local, extract attributes from the local data and set the properties, but if the source is <code>NULL</code> , raise an <code>ORDSYS.ORDSourceExceptions.EMPTY_SOURCE</code> exception; if the source is a <code>BFILE</code> , then extract attributes from the <code>BFILE</code> and set the properties; if the source is neither local nor a <code>BFILE</code> , get the media content into a temporary <code>LOB</code> , extract attributes from the data, and set the properties.

B.2.2.2 Extending Oracle Multimedia to Support a New ORDDoc Data Format

Extending Oracle Multimedia to support a new media data format requires a series of tasks.

Follow these steps to extend Oracle Multimedia to support a new ORDDoc data format:

1. Design your new media data format.
 - a. To support a new media data format, implement the required interfaces in the `ORDX_<format>_DOC` package in the `ORDPLUGINS` schema (where `<format>` represents the name of the new media data format). Use the package body example in [Example B-3](#) as a template to create the package body.
 - b. Then, set the new format parameter in the `setFormat()` call to the appropriate format value to indicate to the media object that package `ORDPLUGINS.ORDX_<format>_DOC` should be called.
2. Implement your new media data format and name it, for example, `ORDX_MY_DOC.SQL`.
3. Install your new `ORDX_MY_DOC.SQL` package into the `ORDPLUGINS` schema.
4. Grant EXECUTE privileges on your new package, for example, `ORDX_MY_DOC`, to PUBLIC.
5. In an application, set the format to `my` to cause your package to be invoked.

A package body listing is provided in [Example B-3](#) to assist you in this operation. Add your variables to the places that say "--Your variables go here" and add your code to the places that say "--Your code goes here".

Example B-3 Package Body for Extending Support to a New ORDDoc Data Format

```
CREATE OR REPLACE PACKAGE BODY ORDX_MY_DOC
AS
  --DOCUMENT ATTRIBUTES ACCESSORS
  PROCEDURE setProperties(ctx IN OUT RAW,
                        obj IN OUT NOCOPY ORDSYS.ORDDoc,
                        setComments IN NUMBER :=0)
  IS
  --Your variables go here.
  BEGIN
  --Your code goes here.
  END;
END;
/
show errors;
```

Related Topics

- [ORDPLUGINS.ORDX_DEFAULT_DOC Package](#)
Use the `ORDPLUGINS.ORDX_DEFAULT_DOC` package as a guide in developing your own `ORDPLUGINS.ORDX_<format>_DOC` media format package.

B.2.3 Supporting Other Video Data Formats

The following subsections describe how to extend `ORDVideo` to support other data formats:

- [ORDPLUGINS.ORDX_DEFAULT_VIDEO Package](#)
- [Extending Oracle Multimedia to Support a New Video Data Format](#)

B.2.3.1 ORDPLUGINS.ORDX_DEFAULT_VIDEO Package

Use the `ORDPLUGINS.ORDX_DEFAULT_VIDEO` package as a guide in developing your own `ORDPLUGINS.ORDX_<format>_VIDEO` video format package. This package sets the `mimeType` field in the `setProperties()` method with a MIME type value that is dependent on the file format.

The `ORDPLUGINS.ORDX_DEFAULT_VIDEO` package is defined as follows:

```
CREATE OR REPLACE PACKAGE ORDX_DEFAULT_VIDEO
authid current_user
AS
--VIDEO ATTRIBUTES ACCESSORS
FUNCTION  getAttribute(ctx IN OUT RAW,
                    obj IN ORDSYS.ORDVideo,
                    name IN VARCHAR2)
            RETURN VARCHAR2;
PROCEDURE setProperties(ctx IN OUT RAW,
                    obj IN OUT NOCOPY ORDSYS.ORDVideo,
                    setComments IN NUMBER := 0);
FUNCTION  checkProperties(ctx IN OUT RAW,obj IN ORDSYS.ORDVideo) RETURN NUMBER;

-- must return name=value; name=value; ... pairs
PROCEDURE getAllAttributes(ctx IN OUT RAW,
                    obj IN ORDSYS.ORDVideo,
                    attributes IN OUT NOCOPY CLOB);

-- VIDEO PROCESSING METHODS
FUNCTION  processCommand(
                    ctx          IN OUT RAW,
                    obj          IN OUT NOCOPY ORDSYS.ORDVideo,
                    cmd          IN VARCHAR2,
                    arguments IN VARCHAR2,
                    result       OUT RAW)
            RETURN RAW;

END;
/
```

[Table B-5](#) shows the methods that are supported in the `ORDPLUGINS.ORDX_DEFAULT_VIDEO` package and the exceptions that are raised if you call a method that is not supported.

Table B-5 Methods Supported in the `ORDPLUGINS.ORDX_DEFAULT_VIDEO` Package

Name of Method	Level of Support
<code>getAttribute</code>	Not supported - raises exceptions: <code>METHOD_NOT_SUPPORTED</code> and <code>VIDEO_PLUGIN_EXCEPTION</code>
<code>setProperties</code>	Supported; if the source is local, extract attributes from the local data and set the properties, but if the source is <code>NULL</code> , raise an <code>ORDSYS.ORDSourceExceptions.EMPTY_SOURCE</code> exception; if the source is a <code>BFILE</code> , then extract attributes from the <code>BFILE</code> and set the properties; if the source is neither local nor a <code>BFILE</code> , get the media content into a temporary LOB, extract attributes from the data, and set the properties.

Table B-5 (Cont.) Methods Supported in the ORDPLUGINS.ORDX_DEFAULT_VIDEO Package

Name of Method	Level of Support
checkProperties	Supported; if the source is local, extract attributes from the local data and compare them with the attribute values of the object, but if the source is NULL, raise an ORDSYS.ORDSourceExceptions.EMPTY_SOURCE exception; if the source is a BFILE, then extract attributes from the BFILE data and compare them with the attribute values of the object; if the source is neither local nor a BFILE, get the media content into a temporary LOB, extract attributes from the media content and compare them with the attribute values of the object.
getAllAttributes	Supported; if the source is local, get the attributes and return them, but if the source is NULL, raise an ORDSYS.ORDSourceExceptions.EMPTY_SOURCE exception; otherwise, if the source is external, raise an ORDSYS.ORDVideoExceptions.LOCAL_DATA_SOURCE_REQUIRED exception.
processCommand	Not supported - raises exceptions: METHOD_NOT_SUPPORTED and VIDEO_PLUGIN_EXCEPTION

B.2.3.2 Extending Oracle Multimedia to Support a New Video Data Format

Extending Oracle Multimedia to support a new video data format requires a series of tasks.

Follow these steps to extend Oracle Multimedia to support a new video data format:

1. Design your new video data format.
 - a. To support a new video data format, implement the required interfaces in the `ORDX_<format>_VIDEO` package in the `ORDPLUGINS` schema (where `<format>` represents the name of the new video data format). Use the package body example in [Example B-4](#) as a template to create the video package body.
 - b. Then, set the new format parameter in the `setFormat()` call to the appropriate format value to indicate to the video object that package `ORDPLUGINS.ORDX_<format>_VIDEO` should be called.
2. Implement your new video data format and name it, for example, `ORDX_MY_VIDEO.SQL`.
3. Install your new `ORDX_MY_VIDEO.SQL` package into the `ORDPLUGINS` schema.
4. Grant EXECUTE privileges on your new package, for example, `ORDX_MY_VIDEO`, to PUBLIC.
5. In an application, set the video format to `my` to cause your package to be invoked.

A package body listing is provided in [Example B-4](#) to assist you in this operation. Add your variables to the places that say "--Your variables go here" and add your code to the places that say "--Your code goes here".

Example B-4 Package Body for Extending Support to a New Video Data Format

```
CREATE OR REPLACE PACKAGE BODY ORDX_MY_VIDEO
AS
```

```

--VIDEO ATTRIBUTES ACCESSORS
FUNCTION  getAttribute(ctx IN OUT RAW,
                    obj IN ORDSYS.ORDVideo,
                    name IN VARCHAR2)

RETURN VARCHAR2
IS
--Your variables go here.
BEGIN
--Your code goes here.
END;
PROCEDURE setProperties(ctx IN OUT RAW,
                    obj IN OUT NOCOPY ORDSYS.ORDVideo,
                    setComments IN NUMBER :=0)

IS
--Your variables go here.
BEGIN
--Your code goes here.
END;
FUNCTION checkProperties(ctx IN OUT RAW, obj IN ORDSYS.ORDVideo) RETURN NUMBER
IS
--Your variables go here.
BEGIN
--Your code goes here.
END;
PROCEDURE getAllAttributes(ctx IN OUT RAW,
                    obj IN ORDSYS.ORDVideo,
                    attributes IN OUT NOCOPY CLOB)

IS
--Your variables go here.
BEGIN
--Your code goes here.
END;
-- VIDEO PROCESSING METHODS
FUNCTION  processCommand(
                    ctx          IN OUT RAW,
                    obj          IN OUT NOCOPY ORDSYS.ORDVideo,
                    cmd          IN VARCHAR2,
                    arguments IN VARCHAR2,
                    result OUT RAW)

RETURN RAW
IS
--Your variables go here.
BEGIN
--Your code goes here.
END;
END;
/
show errors;

```

Related Topics

- [ORDPLUGINS.ORDX_DEFAULT_VIDEO Package](#)
Use the `ORDPLUGINS.ORDX_DEFAULT_VIDEO` package as a guide in developing your own `ORDPLUGINS.ORDX_<format>_VIDEO` video format package. This package sets the `mimeType` field in the `setProperties()` method with a MIME type value that is dependent on the file format.

B.2.4 Supporting Other Image Data Formats

Oracle Multimedia supports certain other image formats through the `setProperties()` method for foreign images.

This method enables other image formats to be recognized by writing the values supplied to the `setProperties()` method for foreign images to the existing `ORDImage` data attributes.

See Also:

The `setProperties()` for foreign images method in *Oracle Multimedia Reference* for more information, and to determine the type of images that are supported this way

B.3 Supporting Media Data Processing

You can extend support for the Oracle Multimedia audio and video object types to include audio and video data processing.

Oracle Multimedia allows the audio and video object types to be extended to support the processing of audio and video data, as described in the following subsections:

- [Supporting Audio Data Processing](#)
- [Supporting Video Data Processing](#)

B.3.1 Supporting Audio Data Processing

Audio data processing is the passing of an audio processing command and a set of arguments to a format package for processing.

To support audio data processing, use the `processAudioCommand()` method. This method is available only for user-defined formats.

See Also:

The `processAudioCommand()` method in *Oracle Multimedia Reference* for a description

B.3.2 Supporting Video Data Processing

Video data processing is the passing of a command and a set of arguments to a format package for processing.

To support video data processing, use the `processVideoCommand()` method. This method is only available for user-defined formats.

 **See Also:**

The `processVideoCommand()` method in *Oracle Multimedia Reference* for a description

C

Oracle Multimedia Sample Applications

This appendix summarizes the Oracle Multimedia scripts and sample applications, which are written in C, SQL, and PL/SQL.

Most of the Oracle Multimedia scripts and sample applications are available after you install the Oracle Database Examples media, which you can download from the Oracle Technology Network (OTN), in the locations shown in the following table.

Table C-1 Oracle Multimedia Sample Applications in Oracle Database Examples Media

Name	Location
ORDImage OCI C	Linux and UNIX: <ORACLE_HOME>/ord/img/demo Windows: <ORACLE_HOME>\ord\img\demo
PL/SQL Web Toolkit Photo Album	Linux and UNIX: <ORACLE_HOME>/ord/http/demo/plsqlwtk Windows: <ORACLE_HOME>\ord\http\demo\plsqlwtk
Code Wizard for the PL/SQL Gateway	Linux and UNIX: <ORACLE_HOME>/ord/http/demo/plsgwycw Windows: <ORACLE_HOME>\ord\http\demo\plsgwycw

This appendix includes these sections:

- [Oracle Multimedia ORDImage OCI C Sample Application](#)
- [Oracle Multimedia PL/SQL Sample Applications](#)

See Also:

Oracle Multimedia on the Oracle Technology Network Web site for additional sample applications that are available for download

C.1 Oracle Multimedia ORDImage OCI C Sample Application

The Oracle Multimedia ORDImage OCI C sample application is written in C.

The Oracle Multimedia ORDImage OCI C sample application shows how to use the Oracle Call Interface (OCI) to access the database and process images.

See Also:

The `README.txt` file in the `demo` directory for requirements and instructions for running the Oracle Multimedia ORDImage OCI C sample application

C.2 Oracle Multimedia PL/SQL Sample Applications

The Oracle Multimedia PL/SQL Web Toolkit Photo Album sample application and the Oracle Multimedia Code Wizard sample application for the PL/SQL Gateway are written in SQL and PL/SQL.

The Oracle Multimedia PL/SQL Web Toolkit Photo Album sample application shows how to upload and retrieve media data using the PL/SQL Web Toolkit and PL/SQL Gateway.

The Oracle Multimedia Code Wizard sample application for the PL/SQL Gateway lets you create PL/SQL procedures for the PL/SQL Gateway to upload and retrieve media data stored in the database using the Oracle Multimedia object types.



See Also:

- [Oracle Multimedia PL/SQL Photo Album Sample Application](#) for more information about installing and using the Oracle Multimedia PL/SQL Web Toolkit Photo Album sample application
- [Oracle Multimedia Code Wizard Sample Application for the PL/SQL Gateway](#) for more information about installing and using the Oracle Multimedia Code Wizard sample application for the PL/SQL Gateway
- The `README.txt` file in the `demo` directory for requirements and instructions for running these Oracle Multimedia sample applications

Glossary

audio data

Media data produced by an audio recorder, an audio source, or by program algorithms. Audio recording devices take analog or continuous signals and convert them into digital values with specific audio characteristics.

codecs

Digital compression and decompression schemes.

content metadata

Data that describes the content of image media, such as the name of the photographer, and the date and time when a photograph was taken.

embedded metadata

Metadata that is stored with image data in the image file format.

heterogeneous media data

Assorted media data, such as audio data, image data, video data, and other types of media data. The data can have a variety of formats, depending upon the application that generated it.

image data

Media data produced by a document or photograph scanner, a video source, other specialized image capture devices, or by program algorithms. Image capture devices take analog or continuous signals and convert them 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.

image interchange format

A well-defined organization and use of image attributes, data, and often compression schemes that enables different applications to create, exchange, and use images. Interchange formats are often stored as disk files.

image metadata format

Standard protocols and techniques used to store image metadata within an image file. Formats include EXIF, IPTC-IIM, and XMP.

lossless compression schemes

Compression schemes that squeeze an image so that when it is decompressed, the resulting image is bit-for-bit identical with the original.

lossy compression schemes

Compression schemes that do not result in an identical image when decompressed, but rather, one in which the changes may be imperceptible to the human eye. Lossy schemes generally provide higher compression than lossless compression schemes.

media data

Data from audio, image, DICOM format medical images and other objects, video, or other heterogeneous media.

metadata

Information about media data, such as object length, compression type, or format.

methods

Procedures that can be performed on objects, such as `getContent()` or `setProperties()`.

Oracle *interMedia*

In Oracle Database 11g Release 1 (11.1), the name Oracle *interMedia* was changed to Oracle Multimedia.

protocols

Image interchange formats exchanged in a sequential fashion over a network.

technical metadata

Data that describes image media in a technical sense, such as the height and width of an image, in pixels, or the type of compression used to store the image.

video data

Media data produced by a video recorder, a video camera, digitized animation video, other specialized video recording devices, or by program algorithms. Some video recording devices take analog or continuous signals and convert them into digital values with specific video characteristics.

Index

A

application development
PL/SQL Gateway feature, [2-15](#)

C

C sample applications
ORDImage OCI C, [C-1](#)
Code Wizard for the PL/SQL Gateway sample
application, [C-2](#)
Code Wizard sample application, [4-2](#)
compression formats
audio, [1-4](#)
image, [1-4](#)
video, [1-13](#)
compression schemes, [1-11](#)
content metadata, [5-1](#)

D

data
loading multimedia, [1-14](#)
data formats, [1-11](#)
database users
default passwords, [A-1](#)
DBA tuning tips, [6-1](#)
digital camera images, [5-2](#)
downgrading an installed version of Oracle
Multimedia, [A-5](#)

E

embedded metadata, [5-1](#)
embedding metadata, [1-11](#)
exception handling
Java, [2-7](#)
PL/SQL, [2-12](#)
EXIF standard, [5-2](#)
extending Oracle Multimedia
audio default format, [B-9](#)
document default format, [B-12](#)
new audio format, [B-10](#)
new data source, [B-6](#)

extending Oracle Multimedia (*continued*)
new media data format, [B-13](#)
new video format, [B-15](#)
video default format, [B-14](#)

I

image file storage standards
EXIF, [5-2](#)
IPTC-IIM, [5-3](#)
XMP, [5-3](#)
image metadata format
defined, [5-2](#)
image watermarking, [1-12](#)
installing Oracle Multimedia, [A-2](#)
interchange formats, [1-11](#)
interchanging metadata, [5-3](#)
IPTC-IIM standard, [5-3](#)

J

Java
exception handling, [2-7](#)
retrieving image properties, [2-5](#)
uploading media, [2-3](#)

L

loading data
multimedia, [1-14](#)
using PL/SQL, [1-14](#)
using SQL*Loader, [1-14](#)
lossless compression, [1-11](#)
lossy compression, [1-11](#)

M

media queries
PL/SQL, [2-11](#)
medical imaging, [1-11](#)
metadata, [5-1](#)
embedding, [5-2](#)
extracting metadata, [1-11](#)
embedding in XML, [3-2](#)

metadata (*continued*)
 embedding metadata, [5-6](#)
 extracting, [1-11](#), [3-2](#), [5-2](#)
 extracting metadata, [5-5](#)
 information about, [5-8](#)
 searching, [3-2](#)
 storing, [3-2](#)
 XML DB, [5-3](#), [5-6](#)
 XML documents, [5-3](#)

metadata examples
 creating a table, [5-4](#)
 embedding metadata, [5-6](#)
 extracting metadata, [5-5](#)

N

news media images, [5-3](#)

O

object relational technology, [1-3](#)
 Oracle interMedia See Oracle Multimedia, [1-1](#)
 Oracle Multimedia, [1-1](#)
 media data storage model, [1-4](#)
 objects types, [1-4](#)
 ORDImage OCI C sample application, [C-1](#)
 ORDPLUGINS.ORDX_DEFAULT_AUDIO
 package, [B-9](#)
 ORDPLUGINS.ORDX_DEFAULT_DOC
 package, [B-12](#)
 ORDPLUGINS.ORDX_DEFAULT_VIDEO
 package, [B-14](#)
 ORDPLUGINS.ORDX_FILE_SOURCE package,
 [B-2](#)
 ORDPLUGINS.ORDX_HTTP_SOURCE
 package, [B-4](#)

P

packages
 ORDPLUGINS.ORDX_DEFAULT_AUDIO,
 [B-9](#)
 ORDPLUGINS.ORDX_DEFAULT_DOC,
 [B-12](#)
 ORDPLUGINS.ORDX_DEFAULT_VIDEO,
 [B-14](#)
 ORDPLUGINS.ORDX_FILE_SOURCE, [B-2](#)
 ORDPLUGINS.ORDX_HTTP_SOURCE, [B-4](#)
 packages or PL/SQL plug-ins, [B-2](#)
 passwords
 installation defaults, [A-1](#)

PL/SQL
 client applications, [2-9](#)
 configuring your environment, [2-10](#)

PL/SQL (*continued*)
 exception handling, [2-12](#)
 loading data, [1-14](#)
 media queries, [2-11](#)
 retrieving media, [2-12](#), [2-14](#)
 uploading media, [2-10](#), [2-14](#)

PL/SQL Gateway feature, [2-15](#)
 PL/SQL packages, [2-14](#)
 PL/SQL sample applications
 Code Wizard for the PL/SQL Gateway, [C-2](#)
 PL/SQL Web Toolkit Photo Album, [C-2](#)
 PL/SQL Web Toolkit Photo Album sample
 application, [3-2](#), [C-2](#)

protocols, [1-11](#)

R

related documents, [xii](#)
 retrieving media
 PL/SQL, [2-12](#), [2-14](#)

S

sample applications
 Code Wizard, [4-2](#)
 downloading from Oracle Technology
 Network, [C-1](#)
 Oracle Multimedia directory locations, [C-1](#)
 PL/SQL Web Toolkit Photo Album, [3-2](#)

SQL*Loader
 loading data, [1-14](#)

T

technical metadata, [5-1](#)

U

upgrading an Oracle Multimedia installation, [A-5](#)
 uploading media
 PL/SQL, [2-10](#), [2-14](#)

V

verifying an Oracle Multimedia installation, [A-5](#)

X

XML
 representing metadata, [5-3](#)
 XML schemas
 registering with Oracle XML DB, [3-2](#), [5-6](#)
 XMP standard, [5-3](#)