Oracle8*i*™

Setting Up the *inter*Media Clipboard, Version 2.1

Oracle Database Server Release 8.1.7 and higher

July 2001

Part Number (TBS)

ORACLE®

Setting Up the *inter*Media Clipboard, Version 2.1, Oracle Database Server Release 8.1.7 and higher

Part Number (TBS)

Primary Author: Chuck Murray

Contributors: Ling Cheng, Dave Diamond, Bob Hanckel, John Janosik, Sharon Lee, Dan Mullen, Mike Rubino

Contents

List of Examples

List of Figures

List of Tables

Send Us Your Comments

Setting Up the *inter*Media Clipboard, Version 2.1

Part Number (TBS)

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

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

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

- Electronic mail: nedc-doc_us@oracle.com
- FAX: 603.897.3825 Attn: *inter*Media Clipboard Documentation
- Postal service:
  Oracle Corporation
  *inter*Media Clipboard Documentation
  One Oracle Drive
  Nashua, NH 03062-2804
  USA

If you would like a reply, please include your name and contact information.

Preface

*Setting Up the inter*Media Clipboard, Version 2.1 describes the installation and configuration operations to enable users to access Oracle *inter*Media data through the *inter*Media Clipboard, Version 2.1 (often referred to in this guide as the Clipboard), which has a Web browser interface.

Intended Audience

This document is intended primarily for people who install and configure software in an Oracle8*i* environment. You are assumed to be familiar with Oracle database concepts, including basic installation and configuration. It is also helpful if you are at least somewhat familiar with Web server software.

This document is not primarily intended for Clipboard end users or administrators, or for end users of Web sites or client applications. (A possible exception is Appendix A, which is intended to help users of the previous Clipboard version make the transition to the current version.) The Clipboard documentation for end users and administrators is provided in the online help.

Structure

This manual contains the following elements:

Chapter 1	Describes user interaction options, the architecture, and an overview of the installation and configuration procedure (with a reference to the online file that contains detailed information).
Chapter 2	Describes the OraDAV parameters that you can specify in <Location> directives in the Apache `httpd.conf` file

Chapter 3	Describes options that you have when creating a container and the system tables and views that OraDAV uses in maintaining containers.
Chapter 4	Describes the programming interface for OraDAV-enabling multimedia content.
Chapter 5	Provides reference information for the procedures in the programming interface.
Appendix A	Describes differences between the previous Clipboard version (Version 1) and the current version. Intended for users of the previous version, which was available from the Oracle Technology Network.

Related Documents

For more information, see the following documents:

- *Oracle interMedia User's Guide and Reference*
- *Oracle9i New Features*
- *Oracle9i Database Administrator's Guide*
- *Oracle9i Database Concepts*

Conventions

The following conventions are also used in this manual:

Convention	Meaning
. . .	Vertical ellipsis points in an example mean that information not directly related to the example has been omitted.
. . .	Horizontal ellipsis points in statements or commands mean that parts of the statement or command not directly related to the example have been omitted
boldface text	Boldface type in text indicates a term defined in the text.
[ ]	Brackets enclose optional clauses from which you can choose one or none.

Documentation Accessibility

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

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

Accessibility of Code Examples in Documentation

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

1

Installation and Configuration

This chapter describes what you need to do to enable users to access and manage interMedia content -- such as audio, video, image, and document files -- in an Oracle database using the *inter*Media Clipboard, Version 2.1 and other convenient Web tools. Providing such flexible and simple access can expand opportunities for your business. For example, employees and customers can access and (if authorized) modify and manage database content using a Web browser or a Microsoft Windows Web folder; they do not need to deal with database connections, SQL statements, or programming.

This chapter includes the following major sections:

1.1 User Interaction Options

When you install and configure the software necessary for the *inter*Media Clipboard, Version 2.1 (often referred to in this document as *the Clipboard*), you enable the following kinds of user interaction with the Oracle database:

- Through a Web browser (version 4 or higher of Netscape or Internet Explorer), using the Clipboard

  Section 1.1.1 describes user interaction through a Web browser.

- Through any DAV-compliant client (such as Microsoft Windows Web folders), in addition to the Clipboard

  Section 1.1.2 describes user interaction through a DAV-compliant client.

For both user interaction options, you must perform installation and configuration operations, as explained in Section 1.3. The options are explained briefly in this section, so that you can better understand what the end users will be experience and what your configuration options are.

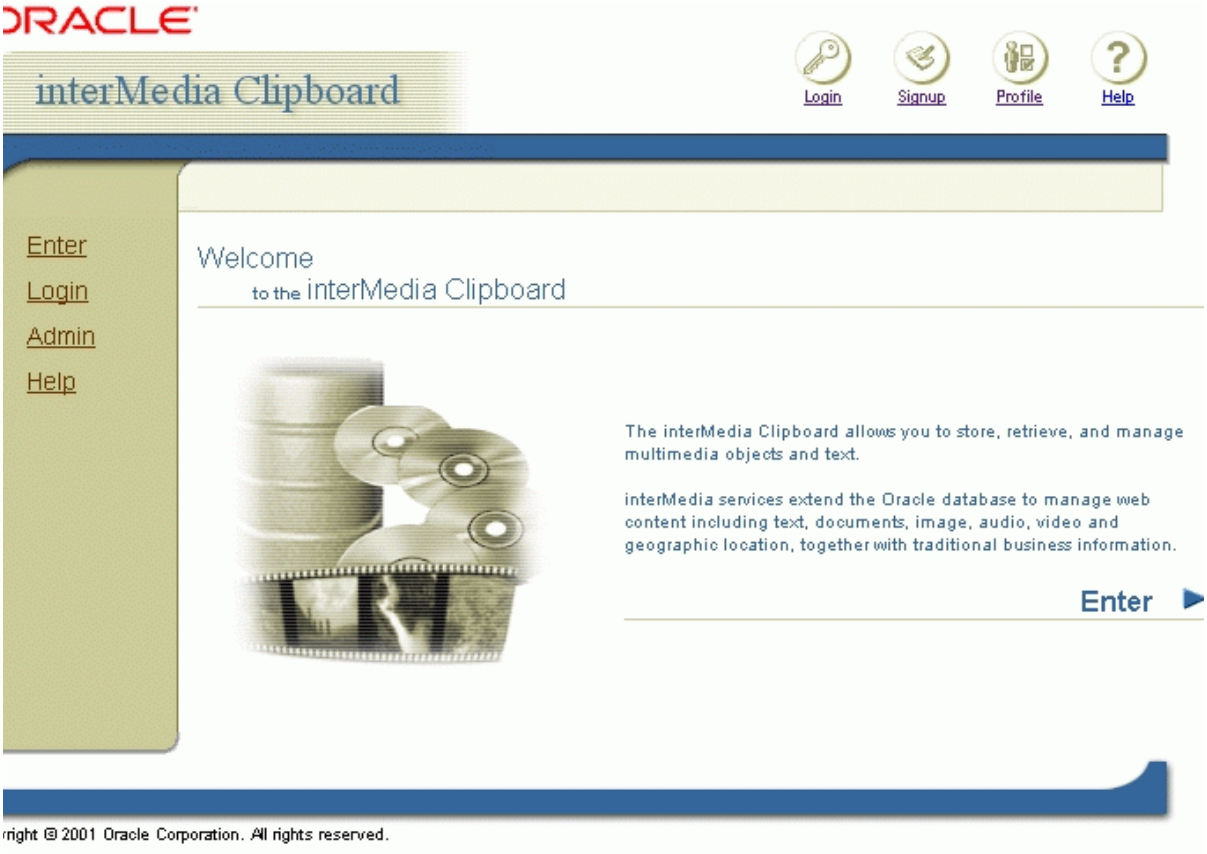
1.1.1 User Interaction Through the *inter*Media Clipboard

When users access the Clipboard (if they are not already logged in), they first see a welcome screen such as the one shown in Figure 1–1, which provides access to all other Clipboard features.

Note: This section provides only an overview of the Clipboard interface. The end-user and administrator documentation for the Clipboard is included in the online help.

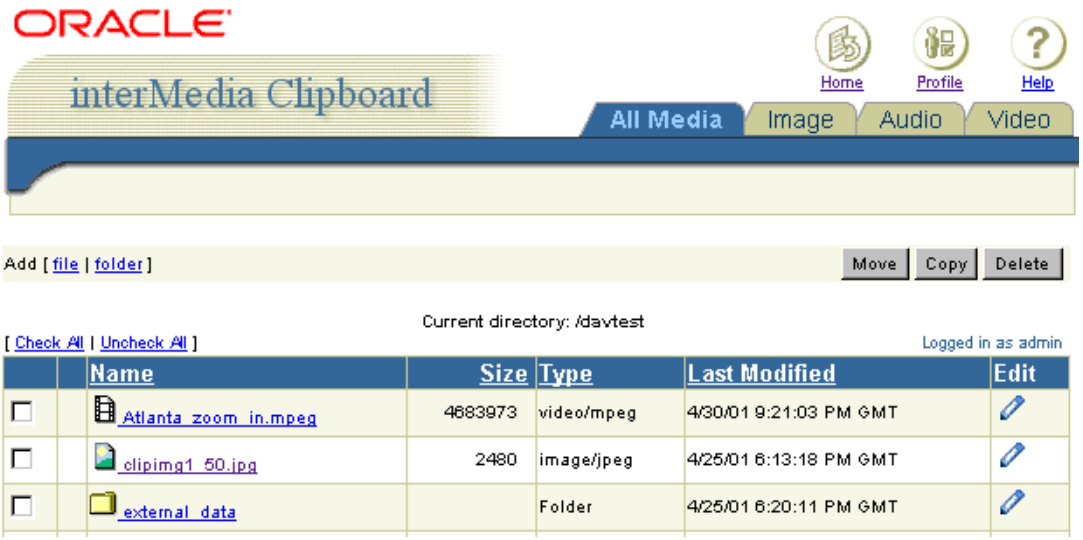
All Clipboard users should click **Help** on any page for detailed explanations of the displays and the available features.

Figure 1–1 interMedia Clipboard Welcome Page



When users click **Enter**, they see the main display of available folders and media files, such as the display in Figure 1–2.

Figure 1–2 interMedia Clipboard Folders and Media Files



In the *inter*Media Clipboard Web browser interface illustrated in Figure 1–2:

- The window displays the contents of a single folder. In this case, the folder is a view of the entire container. (Containers are explained in detail in Chapter 3.)
- Each folder can contain folders and media files. Thus, you can have a multilevel hierarchy of folders: for example, the top-level folder (the view of the entire container) containing a folder, which contains another folder, which contains another folder, and so on (for example, top-level folder containing Folder A, which contains Folder B, which contains Folder C, and so on).
- The icon identifies whether an item is a folder or file: identifies a folder, and other icons identify files of different types. You can control which types of files are displayed by clicking a tab (All Media, Image, Audio, Video) at the top.
- To select a folder or file to perform an operation on it, click the box to the left of its icon and name. (To deselect a selected item, click the box.)

  You can select all items in a folder by clicking Check All, and deselect all items in a folder by clicking Uncheck All.

- To perform an operation on a folder or file, select it, and then click the appropriate command text link or button. Table 1–1 describes common Clipboard operations.

> **Note:** You cannot right-click to perform these operations, because these clicks are interpreted by the Web browser, not the Oracle interface. The browser's right-click menu options (such as Print Frame or Save Link As) depend on the window context and the specific browser.

Table 1–1 Clipboard Operations

To Do This:	Click This:
Add a file to the current folder	Add: file
Add a folder to the current folder	Add: folder
Copy a file or folder (Copying a folder copies all folders and files in and under it in the hierarchy.)	Copy (then specify the destination folder)
Delete a file or folder (Deleting a folder deletes all folders and files in and under it in the hierarchy.)	Delete
Move a file or folder (Moving a folder moves all folders and files in and under it in the hierarchy.)	Move (then specify the destination folder)
Go to a folder	(folder name)
Go up one level in the folder hierarchy (that is, to the folder that contains the current folder)	Go up one level
Rename, control access to, and view the properties of a folder or file	Icon in the Edit column
Read detailed information about the Clipboard and its features (relevant to your current context)	Help
View or modify your user name, password, or display name	Profile (Displays a login box first if you are not logged in.)

Copy and Move have the standard meanings: Copy creates a duplicate but does not remove the original; Move creates a duplicate and does remove the original.

To log in to the Clipboard, click Login (displayed only if you are not currently logged in). If you are not logged in, you can access folders and files that do not have

any controls on their access; however, if you try to access a folder or file that has any access controls, the login box is displayed.

To log out of the Clipboard, exit the Web browser completely or just the current browser window, depending on browser type. For example, with Netscape you must exit the browser completely to log out of the Clipboard, but with Internet Explorer you need only close the current browser window.

To save a file, right-click its name and select the appropriate command from the displayed menu. (The appropriate command, such as Save Link As or Save Target As, is browser-dependent.)

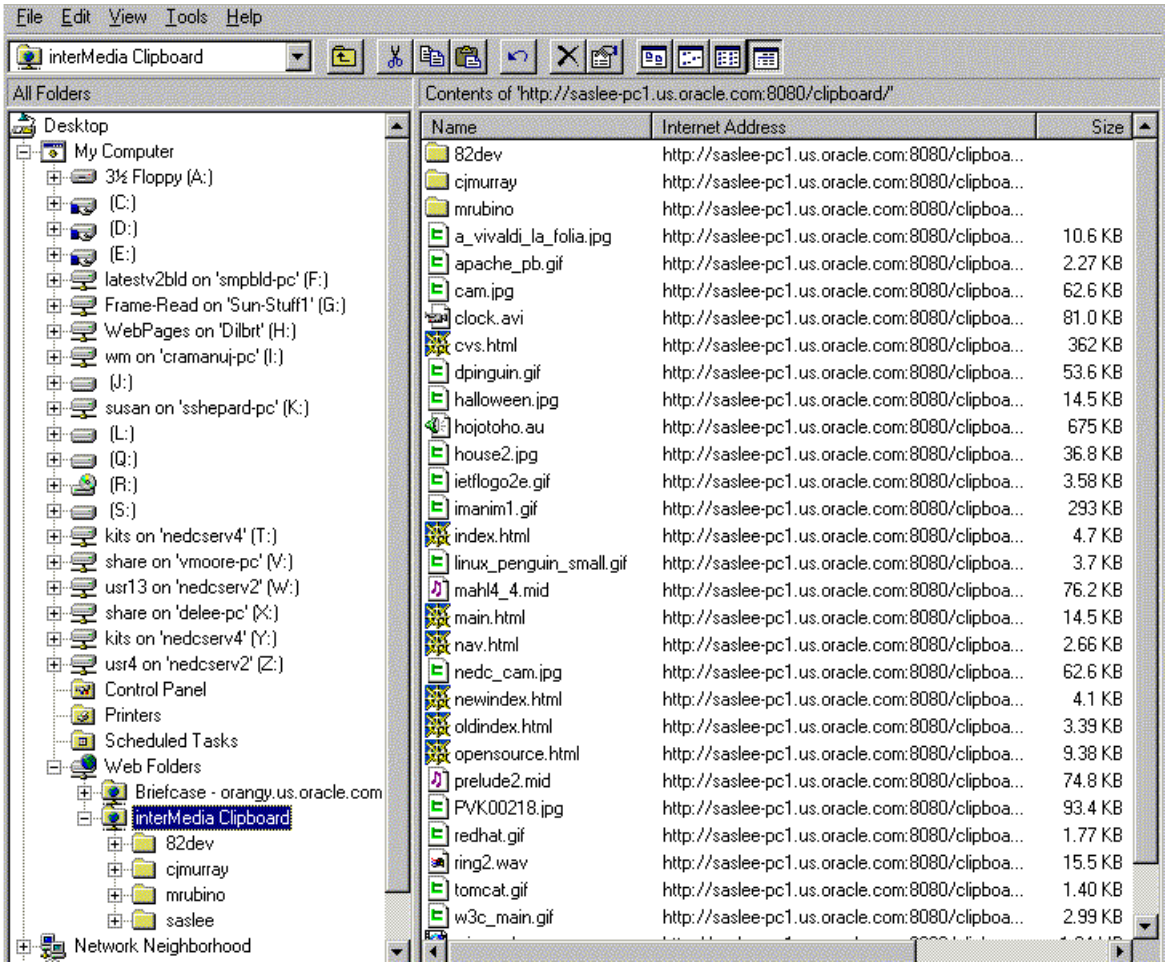
Note: For a detailed explanation of any *inter*Media Clipboard display or option, click **Help**.

1.1.2 User Interaction Through Any DAV-Compliant Client

A **DAV-compliant client** is any application or user interface that complies with the World Wide Web Distributed Authoring and Versioning (WebDAV, or simply DAV) standard. An example of a DAV-compliant client is Microsoft Web folders, which are included in Windows 2000 and can be installed on certain earlier versions of Windows.

Figure 1–3 shows the interface for accessing data through Web folders using Microsoft Windows Explorer.

Figure 1–3 Web Folders Interface



In the Microsoft Web folders interface illustrated in Figure 1–3:

- The same folders and files shown in the Web browser interface (in Figure 1–2) are available, although the order and format of the display depend on the client interface (in this case, Windows Explorer).

- The methods for performing operations (expanding and collapsing folders; selecting items; copying, moving, and deleting items; and so on) depend on the client interface.
- Operations performed using either interface (this one or the Web browser interface) will be reflected in the other interface when the view is refreshed. For example, if you delete a file in the Web folders interface, it will be removed from the Web browser interface display (although you may need to press the Web browser's Refresh or Reload button).

To enable any DAV-compliant client to access the same data available through the *inter*Media Clipboard interface, configure the client to recognize the URL for the OraDAV container. (This URL is described in Section 1.3.3.) For example, with Microsoft Web folders, double-click the **Add Web Folder** icon and specify the URL when the Add Web Folder wizard asks for the location.

1.2 Architecture

The *inter*Media Clipboard and other DAV-compliant clients fit into an architecture in which mod_dav, OraDAV, and Apache JServ within the Apache server communicate with one or more OraDAV containers in one or more schemas in one or more Oracle databases.

A simple form of the architecture is illustrated in Figure 1–4.

Figure 1–4 Architecture for Accessing interMedia Content From the Web

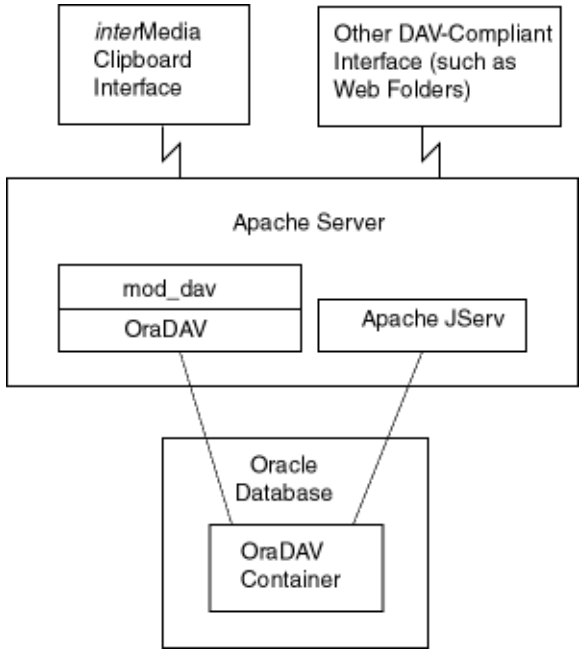


Figure 1–4 shows interaction with a single container (thus involving a single schema) in a single database. However, the configuration options can range in complexity, depending on how varied your data management needs are. Configuration options other than the simple case shown in Figure 1–4 include:

- Multiple containers in a single schema (for example, SCOTT) in a database

  This would allow a database user to manage separate containers, such as for different projects (for example, a construction project container and a transportation / shipping project container).

  The Clipboard servlet can access multiple containers (in the same or multiple schemas) by using Apache JServ zones.

- Multiple schemas, each containing one or more containers

  This would allow multiple database users to manage separate containers.

- Multiple databases, each containing one or more schemas, each having one or more containers

This would allow further flexibility with database users and schemas, as well as replication and failover options.

- Multiple Apache servers on separate systems accessing a single container

  This would allow for high availability of the container, in case of overload or failure involving one of the servlets. It would also provide scalability to handle periods of heavy access.

Before you install or configure the Clipboard, you must decide how simple or complex your database interaction needs are, because these will decide the specific steps you need to take.

1.3 Installation and Configuration Procedure

The detailed installation and configuration instructions are in a text file named `INSTALL` in the *inter*Media Clipboard kit.

If you have not already downloaded the *inter*Media Clipboard kit, download it from the Oracle Technology Network.

To access the Oracle Technology Network, go to

`http://technet.oracle.com`

Select **Software** and then ***inter*Media Plugins and Utilities**, or search for the *inter*Media Clipboard kit.

Carefully read the file named `INSTALL` in that kit, and follows its instructions. The rest of this section summarizes the information in the `INSTALL` file and emphasizes a few considerations.

1.3.1 Prerequisite Software

To install, configure, and use the *inter*Media Clipboard, you must have the following software installed:

- Apache Web server, specifically a version supported by the *inter*Media Clipboard (dependent on the kit name, as explained in the `INSTALL` file)
- mod_dav (dependent on the kit name, as explained in the `INSTALL` file)
- Apache JServ 1.1.2
- Oracle8*i* client release 8.1.7 or higher
- Oracle8*i* database release 8.1.7 or higher

The INSTALL file may contain information about additional software required for using the *inter*Media Clipboard or certain related options.

1.3.2 Overview of the Procedure

To install and configure the Clipboard, you will perform these general steps:

1. Install prerequisite software in an Oracle database, if necessary.
2. Migrate content from Clipboard Version 2.0, if you used that version and have content to be migrated.
3. Create an interMedia container in Oracle.
4. Patch and rebuild mod_dav and Apache for OraDAV support.
5. Install the Clipboard servlet.
6. Configure `httpd.conf` to access Oracle content through OraDAV.
7. Install the annotation facility (optional).
8. Start Apache.
9. Build a DXO (optional, Linux only; enables the use of dynamic loading in Apache).

For detailed information and instructions for each step, as well as other important information, see the `INSTALL` file.

After installing and configuring the Clipboard, you must tell users the URL for accessing it, as explained in Section 1.3.3.

1.3.3 Mapping Containers Under the Root Location

As explained in the `INSTALL` file in the section on configuring `httpd.conf` to access *inter*Media content, you must use the <Location> directive to enable the interMedia Clipboard to access a container. The <Location> directive determines a URL for accessing the container relative to the container's root URL. For example, if the container name is `mywebsite` and the container's root URL is `http://mywebserver`, specifying `<Location /project3>` allows users to access the container by specifying the URL as `http://mywebserver/project3`.

The entry in `httpd.conf` for this example might be:

```
<Location /project3>
DAVDepthInfinity on
DAV Oracle
```

```
DAVParam OraService        MyOracleService
DAVParam OraUser           scott
DAVParam OraPassword       tiger
DAVParam OraContainerName  mywebsite
</Location>
```

As explained in the INSTALL file, though, there are practices to avoid when mapping containers under the root location. Specifically:

- Do not map the root itself. That is, do not specify <Location />.
- Do not map a container as a subelement in the hierarchy to another container. For example, do not specify the following: <Location /project1> and <Location /project1/project2>. However, it is acceptable to specify <Location /project1> and <Location /project2>.

After you define any containers and their mappings, be sure to tell users of the *inter*Media Clipboard the appropriate URL or URLs so that they can access the container contents.

1.3.4 Configuring the Clipboard for Viewing Multiple Containers

You can set up the Clipboard servlet on Apache JServ so that users can access multiple containers, even if the containers are on different databases. By using Apache JServ servlet zones, you can create multiple servlet zones, each of which contains the Clipboard servlet but with different initialization parameters (defined in the clipboard.properties file.)

For example, with servlet zones, Apache JServ can be configured so that when users access a URL like the following:

```
http://mywebserver/clipboard/ct1/oracle.ord.im.clipboard.Navigator
```

they see the container named ct1 on one database; but when they access a URL like the following:

```
http://mywebserver/clipboard/ct2/oracle.ord.im.clipboard.Navigator
```

they see the container named ct2 on another database.

To understand the example in the rest of this section and to follow the instructions, you should understand Apache servlet zones, which are explained at the following location:

```
http://java.apache.org/jserv/zones.html
```

The following example assumes that:

- You have already successfully installed the Clipboard
- Apache JServ has been installed into `/usr/local/jserv`

To set up the Clipboard to point to two different containers, `ct1` and `ct2`, follow these steps:

1. Create two new `zone.properties` files under `/usr/local/jserv/etc`: `zone_ct1.properties` and `zone_ct2.properties`, for the `ct1` container and `ct2` container zones, respectively.

   (These are suggested file names for this example. The actual file names can be anything you choose.)

2. In the zone_ct1.properties file, point the `repositories` parameter to the path of `imclipboard.jar`:

   ```
   repositories=/<path-string>/imclipboard.jar
   ```

   Also, modify the `clipboard_config` parameter to point to the `ct1` zone's `clipboard.properties` file:

   ```
   servlet.oracle.ord.im.clipboard.Navigator.initArgs=clipboard_
   config=/<path-string>/ct1/zone/clipboard.properties
   ```

   (The preceding entry appears to be on two lines in some printed and displayed formats; however, you should type the entry on a single line in the `clipboard.properties` file.)

3. Repeat step 2 with the `zone_ct2.properties` file.

   The path to the `imclipboard.jar` file in the `zone_ct2.properties` file should be the same as in the `zone_ct1.properties` file.

4. In the `jserv.properties` file, add the following under the section titled *Servlet Zones parameters*:

   ```
   zones=ct1, ct2
   ct1.properties=/usr/local/jserv/etc/zone_ct1.properties
   ct2.properties=/usr/local/jserv/etc/zone_ct2.properties
   ```

5. In the `jserv.conf` file, add lines like the following:

   ```
   ApJServMount /clipboard/ct1 ajpv12://localhost:8007/ct1
   ApJServMount /clipboard/ct2 ajpv12://localhost:8007/ct2
   ```

The paths can be anything, as long as they do not conflict with the OraDAV <Location> directives in the `httpd.conf` file or with any other existing `ApJServMount` values. For example, if you have already mounted `/servlet`, you will not be able to mount `/servlet/dir`, because `dir` is a location under an already mounted path.

After Apache is started, users can access the two containers through the Clipboard with the following URLs:

```
http://mywebserver/clipboard/ct1/oracle.ord.im.clipboard.Navigator
http://mywebserver/clipboard/ct2/oracle.ord.im.clipboard.Navigator
```

2

OraDAV Parameters

Before you can access media content using the *inter*Media Clipboard, you must ensure that the OraDAV configuration parameters are set properly. OraDAV parameters, identified by the `DAVParams` keyword, are specified within <Location> directives in the Apache `httpd.conf` file. OraDAV parameters provide:

- A way of configuring how Apache connects to Oracle
- Coarse controls on OraDAV behavior

Table 2–1 lists each OraDAV parameter, whether it is required or optional, and its default value. Section 2.1 provides an example that uses all the parameters, and Section 2.2 discusses each parameter (in alphabetical order) in more detail.

Table 2–1 OraDAV Parameters

Name	Required or Optional	Default Value
ORAService	Required	(none)
ORAUser	Required	(none)
ORAPassword	Required	(none)
ORAContainerName	Required	(none)
ORAPackageName	Optional	ORDSYS.DAV_API_DRIVER
ORAAllowIndexDetails	Optional	FALSE
ORALockExpirationPad	Optional	0 (seconds)
ORAAnnotate	Optional	FALSE

2.1 Example of <Location> Directive

Example 2–1 shows a <Location> directive that uses all of the OraDAV parameters.

Example 2–1 <Location> Directive Using All OraDAV Parameters

```
<Location /oradav>
  DAV oracle
  DAVParam ORAService dbsid.us.oracle.com
  DAVParam ORAUser davuser
  DAVParam ORAPassword davuser
  DAVParam ORAContainerName sales
  DAVParam ORAPackageName ordsys.dav_api_driver
  DAVParam ORAAllowIndexDetails TRUE
  DAVParam ORALockExpirationPad 120
  DAVParam ORAAnnotate TRUE
</Location>
```

In addition to the OraDAV parameters, you may want to consider the value for the `mod_dav` parameter `DAVDepthInfinity`. By default, the `DAVDepthInfinity` value is `off`, because a value of `on` (that is, allowing depth infinity requests) makes it easier for denial of service attacks to occur. However, some clients, such as `sitecopy`, require a DAVDepthInfinity value of `on`.

2.2 OraDAV Parameter Descriptions

This section describes each parameter that was listed in Table 2–1. The parameters are presented in alphabetical order in this section.

2.2.1 ORAAllowIndexDetails

Values: TRUE or FALSE

Default: FALSE

In a non-OraDAV-enabled Apache environment, mod_dav itself does not respond to http `GET` requests. Instead, normal Apache mechanisms are used to respond to `GET` requests. However, when all your content is inside an Oracle database, normal Apache mechanisms cannot be used to respond to `GET` requests and thus OraDAV must respond to `GET` requests.

The `ORAAllowIndexDetails` parameter controls how OraDAV responds when a `GET` request is performed on a DAV collection and no `index.html` file is found in that collection (directory). In a typical Apache environment a separate Apache

module takes control, automatically generating and returning to the client HTML that represents an "index" of the resources (files) in that collection.

An OraDAV-enabled Apache server performs similar actions when responding to a `GET` request on a collection. Additionally, a Description column in the generated index, which contains links to more detailed information about each resource, is generated when ORAAllowIndexDetails is set to TRUE.

These links consist of the URL for the resource itself followed by `?details`.

The default is FALSE, in which case no "Description" column appears in the generated index, and if `?details` is used in a URL, it is ignored and the URL contents are returned.

2.2.2 ORAAnnotate

Values: TRUE or FALSE

Default: FALSE

The `ORAAnnotate` parameter controls the automatic extraction and storage of properties from resources (currently image, audio, and video resources).

If this parameter is set to TRUE, an OraDAV-enabled Apache server performs the following additional actions whenever an image, audio, or video resource is added to a collection:

1. It selects the following from the <ContainerName>$MIME table (described in Section 3.2.1) in the container: the name of the PL/SQL procedure to call to perform the annotation, and annotation level (either summary or detailed).
2. It invokes the annotation routine, passing the following parameters: an XML string (explained in Section 4.4) representing the request context of the HTTP method that the call is serving, the OraDAV document ID for the resource being annotated, the BLOB handle for the resource's data, and the annotation level (S for summary, D for detailed).

The annotation routine is expected to parse attributes from the resource's data and store them as properties in the container's <ContainerName>$PROP table (described in Section 3.2.2). If the annotation level is S, all the properties are concatenated into one XML string and stored as a single property in the <ContainerName>$PROP table. If the annotation level is D, each property is stored as its own row in the <ContainerName>$PROP table.

Several conditions, however, can cause a failure to generate annotations, including the following:

- Unknown or unsupported image, audio, or video formats
- Corrupt image, audio, or video data
- Incorrectly configured annotation routines
- Incorrectly configured Oracle EXTPROC setup (affects image annotation only)

2.2.3 ORAContainerName

Values: (any valid character string, up to 20 characters)

Default: (none: required parameter)

Within the schema specified by the `ORAUser` parameter there must exist a container. (See Chapter 3 for information about creating containers.)

The `ORAContainerName` parameter specifies the name of the container to use for the location.

2.2.4 ORALockExpirationPad

Values: (number of seconds)

Default: 0

The `ORALockExpirationPad` parameter is intended to be used in high-latency network environments, to adjust for the "refresh lock" behavior in Microsoft Office. Microsoft Office attempts to refresh locks on DAV resources just before the lock is set to expire. However, if there is network congestion between the Microsoft Office client and the DAV server, the refresh request might arrive too late, that is, after the lock has expired.

OraDAV periodically looks for locks on resources that have expired and deletes those locks. The `ORALockExpirationPad` parameter can be used to provide some additional ("pad") time between when a lock expires and when that lock is deleted. For example, if ORALockExpirationPad is set to 120, OraDAV does not actually delete locks until at least two minutes after the expiration time.

2.2.5 ORAPackageName

Values: (character string)

Default: ORDSYS.DAV_API_DRIVER

ORAPackageName

The ORAPackageName parameter identifies the OraDAV driver implementation that is to be called when issuing OraDAV commands. The default is the OraDAV *inter*Media driver, which is the ORDSYS.DAV_API_DRIVER package.

2.2.6 ORAPassword

Values: (character string)

Default: (none: required parameter)

The ORAPassword parameter specifies the password associated with the user specified by the ORAUser parameter.

2.2.7 ORAService

Values: (character string)

Default: (none: required parameter)

The ORAService parameter specifies the Oracle database to connect to. The typical format is: <*sid*>.<*domain*>.<*major-domain*>. For example: mydbsid.mydomain.com

2.2.8 ORAUser

Values: (character string)

Default: (none: required parameter)

The ORAUser parameter specifies the database user (schema) to use when connecting to the service specified by the ORAService parameter.

This user must have been granted the following privileges:

- CONNECT
- RESOURCE
- CREATE TABLESPACE
- DROP TABLESPACE
- CREATE ANY TRIGGER

2.3 Other Notes

All OraDAV parameters are passed from Apache to the routines in the ORAPackageName package as part of the context parameter.

The keys are uppercase in Apache (for example, ORAUSER), but the values are not (for example, davuser).

3

Container Options and System Tables

A container is an object that provides a mapping between the interMedia Clipboard display and media content in database tables. Typically, the Clipboard is configured to view a single container, although it can be configured to view multiple containers.

When you create a container, a set of OraDAV system tables and views is created to provide the infrastructure for the container. Options that you choose when creating a container affect the content of these tables and views.

The following SQL procedures are provided for administrators to create and delete containers:

- `orddavcc.sql`
- `orddavdc.sql`

When you run the `orddavcc.sql` procedure to create a container, you are prompted for several options, which are described in Section 3.1.

The tables created by the `orddavcc.sql` procedure are described in Section 3.2.

3.1 Container Options

Table 3–1 lists the container options, in the order they appear in the `orddavcc.sql` procedure.

Table 3–1 Options When Creating a Container

Option	Explanation
ContainerName	Must be a valid SQL object identifier not longer than 20 characters. Example: SALES. This name serves as a prefix for tables, views, triggers, indices, and tablespaces created for the container. Default: `oradav`
ContainerSize	Size in megabytes for the two tablespaces created for the container. 20% is for a tablespace for the OraDAV system tables created for the container. 80% is for the tablespace that owns the default storage for BLOBs. Default: 100 Minimum:10
NoExecute	Specify Y only if you do not want to create the container, but merely want to generate a script (log file) that can be later executed directly in SQL*Plus. Default: N (that is, create the container)
LogFileDir	If you want to log all SQL DDL statements generated to create the container, specify a value for this option. Any value must be a directory owned and known by the Oracle server to which you are connected, and that server must have a `UTL_FILE_DIR=<LogFileDir>` specification in its `INIT.ORA` file. Default: (none, that is, no log file created)
LogFile	When LogFileDir is specified, LogFile is the name of the file to contain all SQL DDL statements generated to create the container. Default: oradav_schema.sql
TraceOn	Enter Y if you want tracing enabled. This means that SQL DDL statements are displayed to serveroutput. To enable this option, you must also enter the following command in SQL*Plus: SET SERVEROUTPUT ON; Default: N
GenIndex	Enter Y or accept the default if you want a small `index.html` file to be preloaded to the container being created. (You can later delete or overwrite the `index.html` file using any DAV client.) This file serves as an easy way to visually verify (using a Web browser) that the container was successfully created. Enter N if you do not want the small `index.html` file to be preloaded. Default: Y

Table 3–1 Options When Creating a Container

Option	Explanation
UseAVI	Enter Y or accept the default to have the multimedia files stored in three separate tables of ORDImage, ORDAudio, and ORDVideo objects, based on the MIME type. Enter N to have all files stored as BLOBs in the xxx$BLOB table. Default: Y

3.2 Container System Tables and Views

The `orddavcc.sql` procedure creates several tables and views that are used by OraDAV to maintain information and perform operations. The table and view names all start with the `ContainerName` option value (see Table 3–1), for example, `SALES$ACE`, `SALES$MIME`, `SALES$PROP`, and so on.

Table 3–2 lists each of these tables and views, with *xxx* representing the value of the `ContainerName` option.

Note: Except for <ContainerName>$MIME and <ContainerName>$PROP, all of these tables and views should be considered *read-only*.

Table 3–2 Container System Tables and Views

Table or View	Description
xxx$ACE	Access control entry.
xxx$ASL	Advanced searching and locating.
xxx$AUDIO	Content storage for ORDAudio files, if the UseAVI container option (see Table 3–1) value is Y.
xxx$BLOB	Content storage, either for files other than ORDAudio, ORDVideo, or ORDImage files if the UseAVI container option (see Table 3–1) value is Y, or for all media files if UseAVI container option value is N.
xxx$CONTAIN	Container identifier.
xxx$CONTAINER	Information used internally to identify, maintain, and track the container.

Table 3–2 Container System Tables and Views (Cont.)

Table or View	Description
xxx$IMAGE	Content storage for ORDImage files, if the UseAVI container option (see Table 3–1) value is Y.
xxx$LOCKS	DAV locks.
xxx$MIME	MIME mapping and content types. Modifications to this table are allowed. For more information about this table, see Section 3.2.1.
xxx$PATH	Hierarchy of path names.
xxx$PRINCIPAL	User names and authorization information.
xxx$PROP	Property information for resources. Modifications to this table are allowed. For more information about this table, see Section 3.2.2.
xxx$RESOURCE	A convenience view that selects certain columns from xxx$PATH and xxx$ASL based on an equijoin of their DOC_ID columns.
xxx$STORAGE	Table used to register storage areas used in partitioning files by MIME type if the UseAVI container option (see Table 3–1) value is Y.
xxx$TBLOB	Temporary table used for generated content.
xxx$VIDEO	Content storage for ORDVideo files, if the UseAVI container option (see Table 3–1) value is Y.

3.2.1 <ContainerName>$MIME Table

The <ContainerName>$MIME table contains information about MIME mapping and content types. Modifications to this table are allowed.

This table contains the columns shown in Table 3–3.

Table 3–3 Columns in the <ContainerName>$MIME Table

Column Name	Data Type	Explanation
EXTENSION	VARCHAR2(30)	A file extension typically used by files of a specified content type. The period must be included in the extension. For example: `.jpg`
PREFERRED_EXT	VARCHAR2(1)	T if EXTENSION is the file extension to be used by OraDAV when generating files of the associated content type; F if EXTENSION is not to be used in this case. Most important when a content type is associated with multiple extensions. For example, if `image/jpeg` is associated with the `.jpg`, `.jpeg`, and `.jpe` extensions, you must specify one with PREFERRED_EXT as T.

Table 3–3 Columns in the <ContainerName>$MIME Table (Cont.)

Column Name	Data Type	Explanation
CONTENTTYPE	VARCHAR2(80)	MIME type typically associated with a file with the extension. For a given CONTENTTYPE, there must be one (and only one) PREFERRED_EXT set to T.
ANNOLEVEL	VARCHAR2(1)	Degree to which data for the content type should be annotated: D (detailed) or S (summary). Ignored if the ANNOROUTINE column is null.
ANNOROUTINE	VARCHAR2(108)	Annotation routine: name of the PL/SQL procedure to call to perform the annotation. Qualify the name of the routine with the schema and package name, if necessary. Make sure the proper grants have been performed so that `ORAUser` can execute the routine. For more information about the annotation routine, see Section 2.2.2.
STORAGE_AREA	VARCHAR2(30)	Storage identifier, corresponding to a STORAGE_AREA column value in the xxx$STORAGE table (described in Table 3–2). For example, if the STORAGE_AREA value for `image/jpeg` is `IMAGE`, then all files of this MIME type, at file creation time, will be inserted and stored in the storage area with storage identifier `IMAGE`.

The ANNOLEVEL and ANNOROUTINE values are not used unless the `ORAAnnotate` parameter is set to TRUE in the <Location> directive.

The following example displays a row from the <ContainerName>$MIME table. (The output is slightly reformatted for readability.)

```
SQL> SELECT * FROM test$mime WHERE extension='.jpg';

EXTENSION PREFERRED_EXT CONTENTTYPE ANNOLEVEL    ANNOROUTINE                    STORAGE_AREA
-------------------------------------------------------------------------------------------

 .jpg          T          image/jpeg     D       ORDSYS.DAVANNOTATE.IMAGEANNOTATE    IMAGE
```

You may want to modify the <ContainerName>$MIME table to do any of the following:

- Add rows to support additional content types
- Change the annotation level from detailed to summary, or vice versa, for specific content types

- Remove the annotation routine for specific content types so that no additional properties are generated for those content types
- Add or change the annotation routine to be called for a particular content type
- Change the storage areas used for newly added objects for certain content types (Existing objects remain in their current storage areas.)

3.2.2 <ContainerName>$PROP Table

The <ContainerName>$PROP table contains information about resource properties. Modifications to this table are allowed.

This table contains the columns shown in Table 3–4.

Table 3–4 Columns in the <ContainerName>$PROP Table

Column Name	Data Type	Explanation
DOC_ID	NUMBER(38)	Internal document identifier. DOC_ID values are unique and never reused.
NAMESPACE	VARCHAR2(1000)	XML namespace associated with this property. Used to avoid ambiguity if there may by duplicate TAG values.
TAG	VARCHAR2(100)	Name assigned to the property. Used with NAMESPACE to identify the property.
LANGUAGE	VARCHAR2(100)	Language in which the property is written.
DESCRIPTION	VARCHAR2(4000)	Additional information (if any) about the property.
DEADORLIVE	VARCHAR2(1)	D for a "dead" property, or L for a "live" property. Dead (static) properties are not directly derived from the document; they are created and maintained by the client. Live (dynamic) properties are derived directly from the document's content; they are maintained by the server, and most cannot be modified by the client.
READONLY	VARCHAR2(1)	Indicates whether the property is read-only: T (True) or F (False).
PROPV_TYPE	VARCHAR2(30)	Data type for the property: VARCHAR2, DATE, NUMBER, or CLOB.
VAL_VARCHAR2	VARCHAR2(4000)	If PROPV_TYPE is VARCHAR2, the value for the property.

Table 3–4 Columns in the <ContainerName>$PROP Table (Cont.)

Column Name	Data Type	Explanation
VAL_CLOB	CLOB	If PROPV_TYPE is CLOB, the value for the property
VAL_DATE	DATE	If PROPV_TYPE is DATE, the value for the property
VAL_NUMBER	NUMBER	If PROPV_TYPE is NUMBER, the value for the property

It is appropriate for writers of annotation routines to make insertions into the <ContainerName>$PROP table.

Note: All existing "live" properties will be deleted before an annotation routine is called, if a document is being overwritten.

4

OraDAV Programming Interface

OraDAV provides PL/SQL procedures that let you **expose** *inter*Media objects: that is, make them visible as files in a container for access through a Web browser. When you expose an object, it is not actually copied into the container; instead, only a reference to the content is inserted into the container. You can execute a procedure within an application to associate one or more URLs with specific files, after which the Clipboard can access the files through those URLs.

These PL/SQL procedures are in a package named DAV_PUBLIC under the ORDSYS schema. By using these procedures, you can expose content from existing tables that contain columns of type ORDImage, ORDAudio, ORDVideo, ORDDoc, or BLOB. (Note: The ORDDoc type is new with Oracle9*i*.)

These tables can be in different schemas from the one containing your OraDAV container, as long as the container's schema is granted the proper access to the tables. (Containers are explained in Chapter 3.)

Note: Do not use any DAV_PUBLIC procedures on any of the container system tables and views described in Section 3.2. Use these procedures only for objects stored in tables other than the container system tables and views.

4.1 Options for Exposing Content

You can expose either just a single object at a time or all objects in a column:

- The Expose_Resource_By_Rowid procedure exposes the single object in a specified column in the row with the associated rowid.
- The Expose_Intermedia_Column and Expose_Blob_Column procedures expose all objects in a specified column.

The following sections describe examples of both options for exposing objects. The reference information for each procedure is in Chapter 5.

4.1.1 Example: Exposing a Single Object

Assume that you work for a real estate firm that wants to make pictures of houses for sale available at its Web site. You need to be able to display the house picture associated with a specific address. Your database and system setup includes the following:

- A container named `real_estate` has been created in schema DAVUSER.
- User SCOTT has a table called HOUSES.
- The HOUSES table has a column called HOUSE_PIX, which of type ORDImage.
- The ORDImage.setProperties() method has been invoked such that the `contentLength` and `mimeType` attributes of the ORDImage objects are set.
- The HOUSES table has a column called HOUSE_ID that uniquely identifies each row.
- User SCOTT has granted SELECT access on HOUSES to DAVUSER.
- An OraDAV-enabled Apache server is running on host `somehost.com`, listening on port 80.
- That Apache server has been configured with an OraDAV-enabled location called `/oradav`.
- The application is connected to the database as DAVUSER.

Example 4–1 extracts information to uniquely identify the desired object and exposes it, associating it with a URL.

Example 4–1 Exposing a Single Object

```
DECLARE
   ldocid INTEGER;
   loblen INTEGER;
   lrowid UROWID;
BEGIN
SELECT gm.rowid,DBMS_LOB.GETLENGTH(gm.media_column)
           INTO lrowid, loblen
           FROM SCOTT.GENERIC_MEDIA gm
           WHERE media_id = 1;

ORDSYS.DAV_PUBLIC.expose_resource_by_rowid (
```

```
   ORDSYS.DAV_PUBLIC.generate_ctx('FOO'),
   'SCOTT',
   'GENERIC_MEDIA',
   'MEDIA_COLUMN',
   'BLOB',
   lrowid,
   '/external_collection',
   'my_file_name',
   '.html',
   'text/html',
   loblen,
   SYSDATE,
   ORDSYS.DAV_PUBLIC.DUPE_MODE_FAIL,
   ldocid);

COMMIT;
END;
/
```

Upon successful execution of the code in Example 4–1, the following URL will return the image from SCOTT.GENERIC_MEDIA where the MEDIA_ID is `1` (assuming that `httpd.conf` contains `<Location /oradav>`):

```
http://somehost.com/oradav/external_collection/my_file_name.html
```

4.1.2 Example: Exposing All Objects in a Column

If you want to expose all objects in a multimedia column and if it seems too tedious to expose the objects one at a time, you can expose all of the objects in the column by using the Expose_Intermedia_Column procedure (for ORDImage, ORDAudio, ORDVideo, or ORDDoc data) or the Expose_Blob_Column procedure (for BLOB data).

Example 4–2 uses the same assumptions as for Example 4–1 in Section 4.1.1. However, Example 4–2 exposes all the images in the HOUSE_PIX column.

Example 4–2 Exposing All Objects in a Column

```
DECLARE
BEGIN
ORDSYS.DAV_PUBLIC.expose_interMedia_column(
   ORDSYS.DAV_PUBLIC.Generate_Ctx('real_estate'), -- Container context string
   'SCOTT',
   'HOUSES',
   'HOUSE_PIX', -- Media column
```

```
   'HOUSE_ID', -- Primary key of table
   NULL, -- No prefix
   NULL, -- No suffix
   NULL, -- Use default parent path.
   1,  -- Add triggers
   DUPE_MODE_FAIL); -- Fail if duplicate would result

COMMIT;
END;
/
```

Upon successful execution of the code in Example 4–1, all house pictures in SCOTT.HOUSES can be returned by URLS that all start with http://somehost.com/oradav/. The remainder of each URL will be the value of the HOUSE_ID column for that house. For example, the following URL will return the image from SCOTT.HOUSES where the HOUSE_ID is `123_main_st`:

```
http://somehost.com/oradav/123_main_st.jpg
```

In this case, it is helpful if the values in the HOUSE_ID column are meaningful and easy to associate with a specific house.

4.2 DAV Methods on Exposed Data

The OraDAV implementation allows some DAV protocol methods on exposed data and disallows other methods, as shown in Table 4–1.

Table 4–1 DAV Methods on Exposed Data

Method	Allowed?
GET	Yes
COPY	Yes, as long as the destination is not an exposed collection
PROPFIND	Yes (but you can restrict massive collections by using `<Limit>` in the `httpd.conf` file)
OPTIONS	Yes
DELETE	Yes (The reference to the data is removed.)
MKCOL	No
PROPPATCH	Yes
PUT	No

Table 4–1 DAV Methods on Exposed Data (Cont.)

Method	Allowed?
LOCK	No
MOVE	No
POST	No
UNLOCK	Yes (but it will always fail because LOCK is not allowed)

If you are using a client on exposed data and a command or operation fails, it may be because the application's command or operation maps to (is implemented using) a DAV method that OraDAV does allow to be used on exposed data.

For detailed information about DAV methods, see RFC2518 (*HTTP Extensions for Distributed Authoring -- WEBDAV*).

4.3 Duplicate File Behavior Options

Procedures that expose a resource or a column have a `dupe_behavior` parameter that controls the behavior when a call to the procedure would result in a duplicate file being added to the collection. For example, if the procedure attempts to expose a file named `emp100id.jpg` and a file named `emp100id.jpg` is already exposed at the specified location (path), the `dupe_behavior` value determines how this condition is handled.

Table 4–2 lists the `dupe_behavior` parameter values that are currently supported.

Table 4–2 dupe_behavior Parameter Values

Value	Explanation
DUPE_MODE_FAIL	Any duplicate file names result in an exception. Note that if `add_triggers` is set to TRUE, the trigger could also generate an exception from a seemingly innocuous SQL INSERT or UPDATE statement. Your application must be prepared to handle these exceptions.

See also the `dupe_behavior` parameter descriptions and the usage notes for the Expose_Blob_Column, Expose_Intermedia_Column, and Expose_Resource_By_Rowid procedures in Chapter 5.

4.4 Context Parameter

A context parameter is required for enabling annotation and for exposing or unexposing data.

- If the `ORAAnnotate` parameter (described in Section 2.2.2) is set to TRUE, the annotation routine is called and passed several parameters, one of which is the request context of the HTTP method that the call is serving.
- The procedures for exposing and unexposing data (documented in Chapter 5) have a required input parameter named `ctx`, which provides the context information for the container in which the data is to be exposed.

This parameter is a string in XML format, an example of which is shown in Example 4–3 (reformatted for readability).

Example 4–3 Context Parameter Passed to Annotation Routine

```
<ORADAV>
  <DAVPARAM>
    <ORACONTAINERNAME>sales</ORACONTAINERNAME>
    <ORALOCKEXPIRATIONPAD>0</ORALOCKEXPIRATIONPAD>
    <ORAPACKAGENAME>ordsys.dav_api_driver</ORAPACKAGENAME>
    <ORAPASSWORD>davuser</ORAPASSWORD>
    <ORASERVICE>djm.us.oracle.com</ORASERVICE>
    <ORAUSER>davuser</ORAUSER>
  </DAVPARAM>
  <REQUEST>
    <OraDAVAPICount>1</OraDAVAPICount>
    <Method>GET</Method><URI>/oradav/nav.html</URI>
    <Accept>image/gif, image/x-xbitmap, image/jpeg, image/pjpeg,
      image/png, */*</Accept>
    <Accept-Charset>iso-8859-1,*,utf-8</Accept-Charset>
    <Accept-Encoding>gzip</Accept-Encoding>
    <Accept-Language>en</Accept-Language>
    <Connection>Keep-Alive</Connection>
    <Cookie>ORA_UCM_INFO=TCP%2fMP%2f8MejanrF0%40Qw%3f%3fCu%2bl%606</Cookie>
    <Host>amigos.us.oracle.com:8080</Host>
    <If-Modified-Since>Thu, 30 Nov 2000 14:49:13 GMT;
        length=2733</If-Modified-Since>
    <Pragma>no-cache</Pragma>
    <User-Agent>Mozilla/4.76 [en] (WinNT; U)</User-Agent>
  </REQUEST>
</ORADAV>
```

Most context parameters do not need to be as detailed as Example 4–3. Because almost every ORADAV routine operates on a container, the context parameter must, at minimum, specify the container name, as shown in Example 4–4.

Example 4–4 Minimal Context Parameter

```
<ORADAV>
  <DAVPARAM>
    <ORACONTAINERNAME>sales</ORACONTAINERNAME>
  </DAVPARAM>
</ORADAV>
```

For many procedure calls, a minimal context parameter string such as Example 4–4 is sufficient. You can use the Generate_Ctx function to generate a minimal context string.

4.5 Other Notes

All OraDAV parameters are passed from Apache to the routines in the `ORAPackageName` package as part of the `context` parameter.

The keys are uppercase in Apache (for example, `ORAUSER`), but the values are not (for example, `davuser`).

5

OraDAV Procedures: Reference

This chapter contains reference information for the procedures in the ORDSYS.DAV_PUBLIC PL/SQL package. The procedures are presented in alphabetical order.

Before you use the procedures in this package, be sure you understand the concepts and guidelines in Chapter 4, which discusses the OraDAV programming interface.

Table 5–1 lists the procedures. The rest of this chapter presents detailed reference information for each procedure.

Table 5–1 OraDAV Procedures

Procedure	Description
Cversion	Returns the container version number (also called the physical version number).
Expose_Blob_Column	Makes objects in a column of BLOB type visible by exposing the content of each object as a file with an associated URL.
Expose_Intermedia_ Column	Makes objects in a column of ORDImage, ORDAudio, ORDVideo, or ORDDoc type visible by exposing the content of each object as a file with an associated URL.
Expose_Resource_By_ Rowid	Makes a single object visible by exposing its content as a file with an associated URL.
Unexpose_Column	Removes the visibility of objects in a column by reversing the effect of the Expose_Blob_Column or Expose_Intermedia_ Column procedure.
Unexpose_Resource_By_ Rowid	Removes the visibility of a single object (associated with a specified rowid) by reversing the effect of the Expose_ Resource_By_Rowid procedure.

Table 5–1 OraDAV Procedures (Cont.)

Procedure	Description
Version	Returns the release (version) number of the DAV_PUBLIC procedures package.

Cversion

Purpose

Returns the container version number (also called the physical version number).

Syntax

```
ORDSYS.DAV_PUBLIC.Cversion() RETURN VARCHAR2;
```

Parameters

None.

Usage Notes

This function returns a number describing the physical version that the *inter*Media OraDAV Driver supports.

Contrast this function with Version, which returns the package version number.

Examples

The following example returns the container version number.

```
SELECT ORDSYS.DAV_PUBLIC.Cversion FROM DUAL;

  CVERSION
----------
       1.3
```

Expose_Blob_Column

Purpose

Makes objects in a column of BLOB type visible by exposing the content of each object as a file with an associated URL.

Syntax

```
ORDSYS.DAV_PUBLIC.Expose_Blob_Column(
   ctx               IN  VARCHAR2,
   schema_name       IN  VARCHAR2,
   table_name        IN  VARCHAR2,
   media_column_name IN  VARCHAR2,
   date_column_name  IN  VARCHAR2,
   mime_column_name  IN  VARCHAR2,
   default_mimetype  IN  VARCHAR2,
   key_column_name   IN  VARCHAR2,
   key_prefix        IN  VARCHAR2,
   key_suffix        IN  VARCHAR2,
   parent_path       IN  VARCHAR2,
   add_triggers      IN  INTEGER,
   dupe_behavior     IN  INTEGER);
```

Parameters

Table 5–2 Expose_Blob_Column Procedure Parameters

Parameter	Description
ctx	Context string for the container on which to operate. (See Section 4.4 for detailed information about the context.)
schema_name	Name of the schema containing the table with the column that is to be exposed.
table_name	Name of the table or view containing the column that is to be exposed. Must not be one of the container system tables or views described in Section 3.2.
media_column_name	Name of the BLOB column that is to be exposed.
date_column_name	Name of the column of type DATE that contains the last-modified date for the data in the BLOB column. If specified as null, SYSDATE (current system date and time) is used.

Table 5–2 Expose_Blob_Column Procedure Parameters (Cont.)

Parameter	Description
mime_column_name	Name of the column in table_name that identifies the MIME type. If specified as null, default_mimetype is used.
default_mimetype	The MIME type to be used in mime_column_name is null or if the data selected from the column is null.
key_column_name	Name of the column in table_name that has unique values to be used as base file names and as the final part of each URL.
key_prefix	Prefix to be used in base file names and the final part of each URL. For example, if key_prefix is emp and key_suffix is id and a row includes a base column value of 100, the resulting file name is emp100id.jpg (if the MIME type is image/jpeg).
key_suffix	Suffix to be used in base file names and the final part of each URL. (Note that the suffix is *not* a file extension such as .jpg or .wav.) For example, if key_prefix is emp and key_suffix is id and a row includes a base column value of 100, the resulting file name is emp100id.jpg (if the MIME type is image/jpeg)
parent_path	Path of the folder to which the column should be mapped. If specified as null, the default is a path in the format: /<schema_name>/<table_name>/<media_column_name>/
add_triggers	0 (do not add triggers) or 1 (add triggers). 0 is the default. If 1 is specified, INSERT, UPDATE, and DELETE triggers are added to table_name so that the Clipboard contents will be automatically changed to reflect actions on the underlying table.
dupe_behavior	Action to be taken if a call to the procedure would result in a duplicate file being added to the collection. For more information and a list of acceptable values, see Section 4.3.

Usage Notes

Before using this procedure, be sure you understand the concepts, guidelines, and examples in Section 4.1.

The following guidelines apply to parent_path:

- If the path already exists, it must point to an empty folder which is not locked. For example, if you specify /external_data/my_blobs, and if my_blobs already exists in the folder external_data, my_blobs cannot contain any files or folders and cannot be locked.

- If the path does not exist, its parent folder must exist and must not be locked. Using the preceding example, if `my_blobs` does not exist, `external_data` must exist and be unlocked.

For information about DAV methods that are allowed and disallowed on exposed data, see Table 4–1 in Section 4.2.

To remove the visibility of a column that had been exposed by this procedure, use the Unexpose_Column procedure.

An alternative to using this procedure is to use the *Expose a BLOB Column* wizard available from the Administration (Admin) page of the Clipboard interface.

Examples

The following example exposes all the images in the HOUSE_PIX column. This example has the same assumptions as in Section 4.1.2, except for the following:

- The column HOUSE_PIX is of type BLOB
- All the BLOB objects in HOUSE_PIX are (JFIF) JPEG files.

```
DECLARE
BEGIN
ORDSYS.DAV_PUBLIC.Expose_Blob_Column(
   ORDSYS.DAV_PUBLIC.Generate_Ctx('real_estate'), -- Container context string
   'SCOTT',
   'HOUSES',
   'HOUSE_PIX', -- BLOB column
   'DATE_LAST_MODIFIED', -- Date column
   NULL, -- No column for MIME type; use default.
   'image/jpeg',
   'HOUSE_ID', -- Primary key of table
   NULL, -- No prefix
   NULL, -- No suffix
   NULL, -- Use default parent path.
   1,  -- Add triggers
   DUPE_MODE_FAIL); -- Fail if duplicate would result

COMMIT;
END;
/
```

Upon successful execution of the code in this example, all house pictures in SCOTT.HOUSES can be returned by URLS that all start with http://somehost.com/oradav/. The remainder of each URL will be the value of the

HOUSE_ID column for that house. For example, the following URL will return the image from SCOTT.HOUSES where the HOUSE_ID is `123_main_st`:

```
http://somehost.com/oradav/123_main_st.jpg
```

Expose_Intermedia_Column

Purpose

Makes objects in a column of ORDImage, ORDAudio, ORDVideo, or ORDDoc type visible by exposing the content of each object as a file with an associated URL.

Syntax

```
ORDSYS.DAV_PUBLIC.Expose_Intermedia_Column(
   ctx               IN  VARCHAR2,
   schema_name       IN  VARCHAR2,
   table_name        IN  VARCHAR2,
   media_column_name IN  VARCHAR2,
   key_column_name   IN  VARCHAR2,
   key_prefix        IN  VARCHAR2,
   key_suffix        IN  VARCHAR2,
   parent_path       IN  VARCHAR2,
   add_triggers      IN  INTEGER,
   dupe_behavior     IN  INTEGER);
```

Parameters

Table 5–3 Expose_Intermedia_Column Procedure Parameters

Parameter	Description
ctx	Context string for the container on which to operate. (See Section 4.4 for detailed information about the context.)
schema_name	Name of the schema containing the table with the column that is to be exposed.
table_name	Name of the table or view containing the column that is to be exposed. Must not be one of the container system tables or views described in Section 3.2.
media_column_name	Name of the column of ORDImage, ORDAudio, ORDVideo, or ORDDoc type that is to be exposed.
key_column_name	Name of the column in table_name that has unique values to be used as base file names and as the final part of each URL.

Table 5–3 Expose_Intermedia_Column Procedure Parameters (Cont.)

Parameter	Description
key_prefix	Prefix to be used in base file names and the final part of each URL. For example, if key_prefix is emp and key_suffix is id and a row includes a base column value of 100, the resulting file name is emp100id.jpg (if the MIME type is image/jpeg).
key_suffix	Suffix to be used in base file names and the final part of each URL. (Note that the suffix is *not* a file extension such as .jpg or .wav.) For example, if key_prefix is emp and key_suffix is id and a row includes a base column value of 100, the resulting file name is emp100id.jpg (if the MIME type is image/jpeg)
parent_path	Path of the folder to which the column should be mapped. If specified as null, the default is a path in the format: /<schema_name>/<table_name>/<media_column_name>/
add_triggers	0 (do not add triggers) or 1 (add triggers). 0 is the default. If 1 is specified, INSERT, UPDATE, and DELETE triggers are added to table_name so that the Clipboard contents will be automatically changed to reflect actions on the underlying table.
dupe_behavior	Action to be taken if a call to the procedure would result in a duplicate file being added to the collection. For more information and a list of acceptable values, see Section 4.3.

Usage Notes

Before using this procedure, be sure you understand the concepts, guidelines, and examples in Section 4.1.

For information about DAV methods that are allowed and disallowed on exposed data, see Table 4–1 in Section 4.2.

To remove the visibility of a column that had been exposed by this procedure, use the Unexpose_Column procedure.

An alternative to using this procedure is to use the *Expose an interMedia Column* wizard available from the Administration (Admin) page of the Clipboard interface.

Examples

The following example (from Section 4.1.2) exposes all the images in the HOUSE_PIX column.

```
DECLARE
BEGIN
ORDSYS.DAV_PUBLIC.expose_interMedia_column(
```

```
   ORDSYS.DAV_PUBLIC.Generate_Ctx('real_estate'), -- Container context string
   'SCOTT',
   'HOUSES',
   'HOUSE_PIX', -- Media column
   'HOUSE_ID', -- Primary key of table
   NULL, -- No prefix
   NULL, -- No suffix
   NULL, -- Use default parent path.
   1,  -- Add triggers
   DUPE_MODE_FAIL); -- Fail if duplicate would result

COMMIT;
END;
/
```

Upon successful execution of the code in this example, all house pictures in SCOTT.HOUSES can be returned by URLS that all start with http://somehost.com/oradav/. The remainder of each URL will be the value of the HOUSE_ID column for that house. For example, the following URL will return the image from SCOTT.HOUSES where the HOUSE_ID is `123_main_st`:

```
http://somehost.com/oradav/123_main_st.jpg
```

Expose_Resource_By_Rowid

Purpose

Makes a single object (associated with a specified rowid) visible by exposing its content as a file with an associated URL.

Syntax

```
ORDSYS.DAV_PUBLIC.Expose_Resource_By_Rowid(
   ctx             IN  VARCHAR2,
   schema_name     IN  VARCHAR2,
   table_name      IN  VARCHAR2,
   value_colname   IN  VARCHAR2,
   value_coltype   IN  VARCHAR2,
   prowid          IN  UROWID,
   parent_path     IN  VARCHAR2,
   base_file_name  IN  VARCHAR2,
   extension       IN  VARCHAR2,
   mimetype        IN  VARCHAR2,
   content_length  IN  INTEGER,
   creation_date   IN  DATE,
   dupe_behavior   IN  INTEGER,
   docid           OUT INTEGER);
```

Parameters

Table 5–4 Expose_Resource_By_Rowid Procedure Parameters

Parameter	Description
ctx	Context string for the container on which to operate. (See Section 4.4 for detailed information about the context.)
schema_name	Name of the schema containing the table with the object that is to be exposed.
table_name	Name of the table or view containing the object that is to be exposed. Must not be one of the container system tables or views described in Section 3.2.
value_colname	Name of the column containing the object that is to be exposed.
value_coltype	Data type of value_colname (for example, BLOB or ORDImage).
prowid	Rowid of the row in table_name to be exposed.

Table 5–4 Expose_Resource_By_Rowid Procedure Parameters (Cont.)

Parameter	Description
parent_path	Path of the folder to which the object should be mapped. If specified as null, the default is a path in the format: /<schema_name>/<table_name>/<media_column_name>/
base_file_name	Name to be used for the base file and in the final part of the URL. For example, if base_file_name is 123_main_street and extension is jpg, the resulting file name is 123_main_street.jpg.
extension	Name to be used for the base file and in the final part of the URL. For example, if base_file_name is 123_main_street and extension is jpg, the resulting file name is 123_main_street.jpg.
mimetype	MIME type of the object. For example: image/jpeg. If specified as null, the MIME type associated with extension is used. If no MIME type is associated with extension, application/unknown is used.
content_length	Length in bytes of the content to be exposed.
creation_date	Creation date of the file. If specified as null, SYSDATE (the current system date and time) is used.
dupe_behavior	Action to be taken if a call to the procedure would result in a duplicate file being added to the collection. For more information and a list of acceptable values, see Section 4.3.
docid	Internal document ID generated by OraDAV.

Usage Notes

Before using this procedure, be sure you understand the concepts, guidelines, and examples in Section 4.1.

For information about DAV methods that are allowed and disallowed on exposed data, see Table 4–1 in Section 4.2.

To remove the visibility of a resource that had been exposed by this procedure, use the Unexpose_Resource_By_Rowid procedure.

Examples

The following example exposes the object in the MEDIA_COLUMN column where the MEDIA_ID value is 1.

```
DECLARE
```

```
   ldocid INTEGER;
   loblen INTEGER;
   lrowid UROWID;
BEGIN
SELECT gm.rowid,DBMS_LOB.GETLENGTH(gm.media_column)
           INTO lrowid, loblen
           FROM SCOTT.GENERIC_MEDIA gm
           WHERE media_id = 1;

ORDSYS.DAV_PUBLIC.expose_resource_by_rowid (
   ORDSYS.DAV_PUBLIC.generate_ctx('FOO'),
   'SCOTT',
   'GENERIC_MEDIA',
   'MEDIA_COLUMN',
   'BLOB',
   lrowid,
   '/external_collection',
   'my_file_name',
   '.html',
   'text/html',
   loblen,
   SYSDATE,
   ORDSYS.DAV_PUBLIC.DUPE_MODE_FAIL,
   ldocid);

COMMIT;
END;
/
```

Upon successful execution of the code in this example, the following URL will return the image from SCOTT.GENERIC_MEDIA where the MEDIA_ID is `1` (assuming that `httpd.conf` contains `<Location /oradav>`):

`http://somehost.com/oradav/external_collection/my_file_name.html`

Generate_Ctx

Purpose

Returns an XML string with minimal context information for the specified container.

Syntax

```
ORDSYS.DAV_PUBLIC.Generate_Ctx(
   container_name  IN VARCHAR2,
   ) RETURN VARCHAR2;
```

Parameters

Table 5–5 Generate_Ctx Function Parameters

Parameter	Description
container_name	Name of the container for which to generate minimal context information. The name is specified by the ORAContainerName parameter (described in Section 2.2.3).

Usage Notes

This function can be used to generate a minimal context string for input to procedures that expose data.

For more information about container context, including examples of the XML string, see Section 4.4.

Examples

The following example returns the container version number. (In the actual output, the entire XML tag is on one line with no space or break.)

```
SELECT ORDSYS.DAV_PUBLIC.Generate_Ctx('real_estate') FROM DUAL;

ORDSYS.DAV_PUBLIC.GENERATE_CTX('REAL_ESTATE')
--------------------------------------------------------------------------------

<ORADAV><DAVPARAM><ORACONTAINERNAME>real_
estate</ORACONTAINERNAME><ORAEXCEPTION>RAISE</ORAEXCEPTION></DAVPARAM></ORADAV>
```

Unexpose_Column

Purpose

Removes the visibility of objects in a column by reversing the effect of the Expose_Blob_Column or Expose_Intermedia_Column procedure.

Syntax

```
ORDSYS.DAV_PUBLIC.Unexpose_Column(
   ctx               IN  VARCHAR2,
   schema_name       IN  VARCHAR2,
   table_name        IN  VARCHAR2,
   media_column_name IN  VARCHAR2);
```

Parameters

Table 5–6 Unexpose_Column Procedure Parameters

Parameter	Description
ctx	Context string for the container on which to operate. (See Section 4.4 for detailed information about the context.)
schema_name	Name of the schema containing the table with the column that is to be unexposed.
table_name	Name of the table containing the column that is to be unexposed. Must not be one of the container system tables or views described in Section 3.2.
media_column_name	Complete directory and file name to use for the resource.

Usage Notes

Before using this procedure, be sure you understand the concepts, guidelines, and examples in Section 4.1.

Examples

The following example (from Section 4.1.1) unexposes the images in the HOUSE_PIX column.

```
DECLARE
BEGIN
```

```
ORDSYS.DAV_PUBLIC.Unexpose_interMedia_Column(
   ORDSYS.DAV_PUBLIC.Generate_Ctx('real_estate'), -- Container context string
   'SCOTT',
   'HOUSES',
   'HOUSE_PIX'); -- Media column

COMMIT;
END;
/
```

Upon successful execution of the code in this example, the images from SCOTT.HOUSES that had been previously exposed will no longer be available through the URL that had been assigned.

Unexpose_Resource_By_Rowid

Purpose

Removes the visibility of a single object (associated with a specified rowid) by reversing the effect of the Expose_Resource_By_Rowid procedure.

Syntax

```
ORDSYS.DAV_PUBLIC.Unexpose_Resource_By_Rowid(
   ctx           IN  VARCHAR2,
   schema_name   IN  VARCHAR2,
   table_name    IN  VARCHAR2,
   column_name   IN  VARCHAR2,
   prowid        IN  UROWID);
```

Parameters

Table 5–7 Unexpose_Resource_By_Rowid Procedure Parameters

Parameter	Description
ctx	Context string for the container on which to operate. (See Section 4.4 for detailed information about the context.)
schema_name	Name of the schema containing the table with the column that is to be unexposed.
table_name	Name of the table containing the column that is to be unexposed. Must not be one of the container system tables or views described in Section 3.2.
path	Complete directory and file name to use for the resource.
prowid	Rowid of the row in table_name to be unexposed.

Usage Notes

Before using this procedure, be sure you understand the concepts, guidelines, and examples in Section 4.1.

Examples

The following example (from Section 4.1.1) unexposes the image in the HOUSE_PIX column where the HOUSE_ID value is 123_main_st.

```
DECLARE
   v_rowid UROWID;
BEGIN
SELECT T.ROWID, INTO v_rowid FROM SCOTT.HOUSES T
   WHERE T.HOUSE_ID = '123_main_st';

ORDSYS.DAV_PUBLIC.Unexpose_Resource_By_Rowid(
   ORDSYS.DAV_PUBLIC.Generate_Ctx('real_estate'), -- Container context string
   'SCOTT',
   'HOUSES',
   'HOUSE_PIX', -- Media column
   v_rowid);

COMMIT;
END;
/
```

Upon successful execution of the code in this example, the image from SCOTT.HOUSES with HOUSE_ID = '123_main_st' that had been previously exposed will no longer be available through the URL that had been assigned.

Version

Purpose

Returns the version (release) number of the DAV_PUBLIC procedures package.

Syntax

```
ORDSYS.DAV_PUBLIC.Version() RETURN VARCHAR2;
```

Parameters

None.

Usage Notes

Contrast this function with Cversion, which returns the container (physical) version number.

Examples

The following example returns the DAV_PUBLIC procedures package version number.

```
SELECT ORDSYS.DAV_PUBLIC.Version FROM DUAL;

VERSION
--------------------------------------------------------------------------------

OraDAV V1.2
```

A

Comparison with Previous Version

This appendix is for users of the previous version of the *inter*Media Clipboard (Version 1), which was made available on the Oracle Technology Network. If you are familiar with the Version 1 interface, this appendix will help you to make the transition to Version 2.1.

Table A–1 compares how certain features are implemented and supported in the two versions.

Table A–1 Differences Between Clipboard Versions 1 and 2.1

Feature	Version 1	Version 2.1
Application type and platforms	Client application; available only on Windows NT.	Web browser interface; no operating system restriction.
Installation	Requires one installation per user.	One installation for multiple users.
Drag and drop	Drag multimedia objects from an Oracle8i database to a Web authoring tool. When you drag and drop multimedia objects from a database into a Web authoring tool, the Clipboard generates the necessary URLs to retrieve an object from the database.	Drag and drop is not supported through the Clipboard browser interface. Use File upload. Note that drag and drop is supported if users access Clipboard data through Web Folders. URL generation is no longer needed.
Capture from files	Capture multimedia objects from files and store them in the database.	Upload files using the Clipboard browser interface.
Capture from URLs	Capture multimedia objects from URLs and store them in the database.	Upload files from URLs using the Clipboard browser interface.

Table A–1 Differences Between Clipboard Versions 1 and 2.1 (Cont.)

Feature	Version 1	Version 2.1
Capture from external sources	Capture image objects from external sources, such as cameras and scanners, and store them in the database.	Not supported natively; use file upload.
Edit and reload	Edit multimedia objects with your favorite editor and reload the updated object into the database.	Not supported natively; use file upload.
Database connection	Connect to database using Database Agent.	Connect to any database and schema specified by Clipboard configuration.
Authentication	Database Agent.	Complete security model for protecting individual objects: authentication / authorization, plus access control entries available for all folders and files.
Annotations	Not supported.	Annotation of multimedia objects done automatically at file upload time; annotations can be viewed.
Sorting	Sort order for table.	Sort folders by type, size, date, name.
Selection query	Enter selection query.	Not supported for this release.
Deleting	Delete row.	Delete file. (Deleting objects from Clipboard storage areas is supported, but not deleting objects from external tables that have been exposed.)
Moving	Not supported.	Move folders or files (but cannot move exposed objects).
Copying	Not supported.	Copy folders and files. (Copying an exposed object copies the media data into a Clipboard storage area.)
Displaying	Display multimedia objects in the Clipboard application.	Display images and videos and play audio in the browser interface.
Saving	Save the database object to a local file.	Download the file using the Web browser.

Table A–1 Differences Between Clipboard Versions 1 and 2.1 (Cont.)

Feature	Version 1	Version 2.1
Organizing	Organize data by database tables.	Organize data hierarchically with folders created by users.
Inserting	Insert a row into a table.	Upload a new file. (Inserting objects into Clipboard storage areas is supported, but not inserting objects into external tables that have been exposed.)
Coding	Code wizard for creating PL / SQL procedures to retrieve and store multimedia objects in the database and to set attributes of the object.	No code wizard necessary:· URLs are mapped automatically for retrieving multimedia objects, and annotations of objects are done automatically at file upload time.

For more information about DAV methods supported on exposed data, see Section 4.2.

Index

S

T

U

V

W

Z