O_2 System Administration Reference Manual

Release 5.0 - May 1998

Who should read this manual

This manual contains a comprehensive list of O_2 administration commands. The full syntax and all options are detailed.

It should be read in conjunction with the *O_2 System Administration Guide*.

- ***O_2 System Administration Guide***

This manual is for the O_2 system administrator. It outlines the elements which comprise the O_2 package and describes how to install O_2. It also describes how to customize and tune the system for individual users. The manual describes all configuration options and the utility programs needed for initializing, safe-guarding, restoring, and deleting named O_2 systems.

This manual should be read in conjunction with the *O_2 System Administration Reference Manual* which contains a full list of administration commands.

Other documents available are outlined, click below.

See O2 Documentation Set

TABLE OF CONTENTS

This manual is divided into the following chapters:

TABLE OF CONTENTS

1 Introduction

O_2 SYSTEM ADMINISTRATION COMMANDS

O_2 provides a number of system administration functions. These functions are called through commands which are described in this manual. The context in which these functions may be used and the best way to get the most from an O_2 system is discussed in the *O_2 System Administration Guide*.

Administration commands fall into three categories. There are O_2 system commands, O_2 database administration commands and O_2 schema administration commands.

System commands include those necessary for starting and stopping O_2, copying, checking, removing and reformatting, among others.

O_2 database administration commands are known as O_2DBA commands and they include commands to administer volumes, bases, indexes, clusters and so on.

O_2 schema administration commands are known as O_2DSA commands and they include commands to create a schema or to export some of its classes, for example.

If you are using O_2C refer to the O_2C documentation for more specific commands.

1 Introduction

Structure of this manual

O_2 system commands are outlined in Section 2. Commands are given in alphabetical order.

O_2DBA and O_2DSA commands are described in Sections 3 and 4. Each section is divided into subsections with a subsection per element that can be administered given in alphabetical order. For example applications, bases, etc.

The actions which can be carried out on each element are listed in alphabetical order. For example, create, delete, display, etc.

This manual contains the following sections:

Which commands can you use?

Regardless of the type of license you have, you can use all system commands and O_2DBA commands.

To use O_2DSA commands you need a development licence.

General use of commands

O_2 commands are case sensitive and multiple commands must be separated by semicolons. However, the system ignores spacing or line breaks. In the descriptions below spacing is for presentation purposes only.

You can type the commands directly in O_2 shell, O_2DBA or O_2DSA.

O_2 executes a command or series of commands when it encounters a termination indicator. This is **control/D** for Unix and **control/Z** for Windows.

You can give comments between **/*** and ***/** (as in C).

Help

To get on-line help, use the command:

```
help [O2_keyword]
```

If you type **help**, you obtain a full list of O_2 commands. The syntax and a brief description is given for each command.

When you use an *O2_keyword*, **help** gives information for those commands which contain the keyword.

Batch files

Commands may be executed from a file instead of being entered manually. You do this by typing the command:

# "/path/filename" or **#include "/path/filename"**

The **path** can be any valid path. It can begin with the O_2 keyword **O2HOME**, which indicates the O_2 installation directory. All commands in **filename** must be separated by a ";".

You can also execute commands from a file by redirecting the standard input as follows: **o2dba_shell <filename**. All the commands in **filename** must be separated by a “;”.

For the redirection to work properly, the last command in **filename** must be quit. If it is not, an implicit abort is carried out after the last command.

Presentation conventions

The following presentation conventions are used in this manual.

Syntax presentation

O_2 commands are presented in a **typewriter** type face as are all the brackets, commas and colons that are part of the language syntax, e.g:

```
import schema o2kit class Date, Bitmap;
```

Keywords are presented in typewriter typeface while other items are in italics. These include information you must enter and terms which summarize another part of the command. Options are enclosed in square brackets.

For example:

```
[create] cluster root_name on path
```

If the closing square bracket is followed by an ellipsis (...) then you can repeat the information entered.

```
export schema [class Class_name [, Class_name]...]
```

Without the ellipsis, the information can be entered only once.

```
display volume [volume_name] [stat]
```

When different values can be input they are shown in braces { } and each choice is separated by the word *or*.

```
catalog transaction {on or off}
```

When braces, square brackets and the word **or** are part of the command syntax they are not shown in italics but in typewriter typeface,

`{ }, [ ], or`.

Naming conventions

Names in O_2 must begin with a letter and can be followed by any number of letters, digits or underscore characters. You can use upper-case and lower-case letters but names are case-sensitive.

- ***Names***

These are *class_name, attribute_name*, *method_name*, *property_name*, named value or named object (*obj_or_value_name*), *volume_name*, *schema_name*, *base_name*, *constant_name*.

There are several other special kinds of names:

- ***user_name***

This is the operating system log-in name.

- ***"/path/filename" and "/directory_path"***

These specify an operating system file and directory name. They must be valid paths and must be enclosed in double-quote marks. They are referred to as the *path_name* in the commands in this manual.

- ***documentation***

This is free text that can have multiple lines.

Set up environment

A number of commands exist to set up your environment.

An O_2 environment consists of the following elements:

- Current volume
- Current schema
- Current base
- Current class

The **display** command shows the current environment.

An O_2 named system is a logical unit made up of one or more physical volumes, one or more schemas, and one or more bases. There may be several named systems active on one computer or network, but a single O_2 session operates on one named system, which must be specified when O_2 is started.

The name of the base, schema, volume or class is often optional in a command. This is because a default value is taken. If the name is not entered, the system takes the current base, the schema name as established by the **set base** or **set schema** command, or the current volume, initially **DefaultVol**.

DefaultVol is the first user volume, created during system initialization. The current class is defined using the **set class** command.

O_2 automatically changes the current volume whenever a **set schema** or **set base** command specifies a schema residing in a different physical volume. The current volume always follows the current schema.

There is no **set volume** command.

Within a named system, the relationship between physical volumes, logical bases and schemas is recorded in a catalogue. Commands which create, delete or rename these elements involve changes to the catalogue.

Updating the catalogue can be made part of a development transaction which you commit explicitly. Alternatively, you can carry out each catalogue update as a transaction. In this way O_2 commits updates immediately.

Which method you use is defined by the following command:

```
catalog transaction {on or off}
```

where **off** is the default value.

2

O_2 System Commands

This chapter outlines the O_2 system commands. It is divided into the following sections :

- Options for the O2 System Commands
- O2 System Commands

The programs called by these commands are found in the **bin** subdirectory of the O_2 installation directory.

O2 System Commands

2.1 Options for the O_2 System Commands

All O_2 system commands need options to run. There are two kinds of options:

- Options for O_2Store clients - these are the o2server options.
- Options for O_2 clients - these are the o2 options.

2.1.1 o2server options

These options are used by **o2server**, **o2dba_init**, **o2dba_monitor**, **o2dba_shutdown**, **o2dba_backup, o2dba_restore, o2dba_copy**, **o2dba_env**, and **o2dba_rm**.

The options are found by descending order of priority:

- In the command line.

  Example:

  ```
  o2server -mode single-user
  ```
- As a string in the O2SERVEROPTIONS environment variable.

  Example:

  ```
  setenv O2SERVEROPTIONS "-mode single-user"
  ```
- In the O2HOME/ .o2serverrc configuration file.

  Example:

  ```
  my_system.mode = single-user
  ```

  Refer to the *O_2 System Administration Guide* for further details concerning the configuration file .o2serverrc.

An alphabetical list of all the options that you can define in the .o2serverrc file is as follows:

+[system.] **background**

[system.] **cachesize =** size in kilobytes (server cache)

[system.] **cataldir =** path to directory

[system.] **coldlogdir =** path to directory

[system.] **defaultvolsize =** size in kilobytes

+[system.] **foreground**

[system.] **logdir =** path to directory

[system.] **logsize =** size in kilobytes

[system.] **mode =** multi-user, single-user, or transaction-off

[system.] **server =** machine name for o2server

[system.] **shadowdir =** path to directory

[system.] **shadowsize =** size in kilobytes

system = system name

+[system.] **verbose**

2.1.2 o2 options

These options are used by **o2shell**, **o2dsa_shell**, **o2dba_shell**, **o2tools**, **o2dba_schema_dump**, **o2dba_schema_load**, **o2dba_check**, and all C++ system commands (**o2ccp_**, refer to the *C++ Binding Reference Manual*). For the C++ Runtime options see both C++ manuals, the *C++ Binding Reference Manual* and the *C++ Binding Guide*. For the OQL optimizer option refer to the Indexing Chapter in the *System Administration Guide*.

The options are found by descending order of priority:

- In the command line.

  Example:

  o2dba_check -verbose

- As a string in the O2OPTIONS environment variable.

  Example:

  setenv O2OPTIONS "-verbose"

- In the HOME/ .o2rc configuration file that is defined in the user HOME.

  Example:

  +my_system.verbose

- In the O2HOME/ .o2rc configuration file.

  Example:

  +my_system.verbose

The .o2rc configuration file

The .o2rc configuration file may exist in any user HOME. A global file can also exist in the O2HOME directory. This file contains free-formatted lines, where each line defines an option applied to an O_2 client.

The general syntax of an option definition is:

[system_name.] property_name = property_value

or

+[system_name.] boolean_property_name

Example:

```
my_system.cachesize = 400000
+ my_system.verbose
```

The system_name may be omitted. In this case the property applies to the system used by the O_2 command. This can be seen in the following example:

```
system = my_system
cachesize = 40000
another_system.cachesize = 20000
```

A line beginning with the "**;**" character is a comment.

An alphabetical list of all the options that you can define in the .o2rc files is as follows:

Options for the O2 System Commands :

+[system.] **alpha** disable the graphical option for browsing

[system.] **cachesize =** size in kilobytes (client cache)

[system.] **libpath =** path1: path2: ...pathn (directories for dynamic libraries)

[system.] **libs =** lib1: lib2: ..libn (dynamic libraries for O_2C)

+[system.] **o2cccheck**[1] do dynamic type checking in C++ applications

[system.] **o2ccnumber**[2] = number of C++ objects in the erase buffer

[system.] **oql_index_selectivity**[3] = minimum selectivity for choosing in index

[system.] **server =** machine name where o2server runs

[system.] **swapdir =** path to directory

[system.] **swapsize =** size in kilobytes

system = system name

+[system.] **verbose**

[1] Refer to the *C++ manuals*

[2] Refer to the *C++ manuals*

[3] See Indexing in the *System Administration Guide*

2.2 O_2 System Commands

This chapter outlines the commands listed below. They are given in alphabetical order.

- o2api_shell
- o2dba_backup
- o2dba_backup_display
- o2dba_backup_label
- o2dba_base_dump
- o2dba_base_load
- o2dba_check
- o2dba_copy
- o2dba_env
- o2dba_init
- o2dba_monitor
- o2dba_move_volume
- o2dba_restore
- o2dba_rm
- o2dba_schema_dump
- o2dba_schema_load
- o2dba_shell
- o2dba_shutdown
- o2dba_shell
- o2patch
- o2server
- o2shell
- o2tools

o2api_shell

Summary

Calls up the O_2 Schema administration utility, which allows you to access the O_2Engine API development commands.

Syntax

`o2api_shell` [**`-system`** *`system_name`*] [**`-server`** *`machine_name`*]

`o2api_shell -version`

`o2api_shell -help`

`o2api_shell -env`

Description

This system command allows you to develop and test O_2 applications intended to be used exclusively with O_2Engine API.

It gives you access to all the O_2 commands, as does **`o2shell`**. Unlike **`o2shell`**, however, no source support is provided.

For instance, if you create a class with O_2Engine API primitives you can display this class with **`o2api_shell`**, while **`o2shell`** will fail because it expects to find the source code of the class.

Options

`-system`	Specifies the O_2 named system to connect to. An **`o2server`** must already be running on this system before **`o2api_shell`** is invoked.
`-server`	Specifies the machine name where **`o2server`** is running.
`-version`	Displays information about the current version and exits.
`-help`	Displays information about the possible options and exits.
`-env`	Displays the actual value taken for the options and exits. Allows you to verify that the option mechanism (explicit and implicit) is correctly set.

Default Options

The previous o2options and others can be implicitly set as explained in 2.1.2.

Environment variables

`O2HOME`	Is mandatory and contains the path to the O_2 installation directory.
`O2OPTIONS`	See 2.1.2.

2 O2 System Commands

o2dba_backup

Summary Backs up a system

Syntax

```
o2dba_backup{-file filename [ -size filesize ] |
             -tape device_name}
[-system system_name] [-server server_name]
[-witharchive] [-archivelog] [-nointeractive] [-continue]

[-date [yy-]mmdd[HHMM]
[-period {num[-HHMM] | numH | dd[-HHMM]W | dd[-HHMM]M]}
[-verbose]

o2dba_backup -version

o2dba_backup -help

o2dba_backup -env
```

Description

o2dba_backup performs system backup either on-line or off-line depending on whether the server is running.

By default, **o2dba_backup** carries out a full backup of the system. The server's log mode (hot or cold) is left unchanged after the backup. If option **-witharchive** is invoked, a full backup of the system is followed by continuous backup of the log files. In this case **o2dba_backup** sets the server to cold log mode.

If **o2dba_backup** is invoked with option **-archivelog**, the cold log files are continuously archived. The server is set to cold log mode. This option is similar to the option **-witharchive** except that a full system backup is not carried out.

The physical supports for the backup can be either tapes or regular files on separate physical devices. The backup can extend across multiple backup media volumes. If the backup device is a regular file the option **-size** specifies the maximum file size in kilo-bytes. There are three conditions in which **o2dba_backup** stops and waits for a new backup file: the physical device is full; the maximum size of a file allowed by the operating system is reached; or the maximum size of the file specified by the option **-size** is reached.

A unique backup *identifier* is generated and written to the first volume of the backup media when a backup session begins. The *identifier* is composed of a time-stamp of the session and the name of the system being saved. When restoring the system, this identifier can be used to identify the desired backup version.

By default, **o2dba_backup** runs in interactive mode. If a backup spans multi-volumes the system queries the user for backup media volume changes. If option **-nointeractive** is invoked, the system sends a message to the backup history log file, and polls periodically the backup device until a volume change has been detected, upon which it resumes system backup. In both modes, **o2dba_backup** verifies that the mounted backup media volume works. The option **-nointeractive** is useful when **o2dba_backup** is integrated into an automatic system administration environment.

Entries are generated into the backup history file. These are:

- entries of the beginning and the end of the backup session;
- entries about backup media volumes and their log files*;*

The above information is used when **o2dba_backup** resumes system backup after having been stopped.

By default, **o2dba_backup** begins a new backup session. To resume the preceding backup session instead of starting a new one use option **-continue**. To resume the preceding system backup session after a server shutdown or a system crash, **o2dba_backup** requires the first volume of the backup media in order to retrieve the *identifier* of the session. **o2dba_backup** must scan the media volumes to locate the volume to continue the backup.

Signal **INT** or **TERM** are used to terminate a system backup session, which leaves the server in cold log mode. Afterwards, **o2dba_backup** can be re-launched to either begin a new backup session or resume the preceding session with option **-continue**. The signals **INT** and **TERM** are utilized when backup is carried out by a third party backup product, for further information see the *System Administration Guide*.

Options

-archivelog Archive cold log files without doing system backup. The program continues to run until explicitly stopped.

-continue Resumes the preceding backup instead of beginning a new one. You can change backup type by giving another date or period by samples.

-date [yy-]mmdd[HHMM]

The backup should be performed up to a given time.

H	hour - 00 to 23
M	minute - 00 to 59
d	day of month - 01 to 31
m	month of year - 01 to 12
y	year - 00 to 99

Note: 00 corresponds to the year 2000.

-file The name of the file which will contain the backup.

-period *period*

Begins a new backup session periodically according to the rule defined by *period*. The running session is terminated before the new session begins. This guarantees that information can not be lost between two successive backup sessions. Argument *period* must have one of the following forms:

num [-HHMM]

Begins a new session every *num* days;

num	number of days
H	hour - 00 to 23
M	minute - 00 to 59

*num*H

Begins a new session every *num* hours;

num	number of hours

dd [-HHMM]W

Begins a new session every week*;*

dd	days of the week - 01 to 07
H	hour - 00 to 23
M	minute - 00 to 59

dd [-HHMM]M

Begins a new session every month*;*

dd	days of month - 01 to 31
H	hour - 00 to 23
M	minute - 00 to 59

-server Run the **o2server** process on *server_name*, if not already running.

-system Specifies the O_2 system name. If this option is not used, the system name given by the environment is used (see Default options below).

-tape The name of the device which will contain the backup.

-witharchive The system backup is followed by cold log file archiving. The program continues to run until explicitly stopped.

-verbose Returns additional information about the operation.

-version Displays information about the current version and exits.

-help Displays information about the possible options and exits.

-env Displays the actual value taken for the options and exits. Allows you to verify that the option mechanism (explicit and implicit) is correctly set.

Default Options The previous o2 server options as well as other options can be implicitly set as explained in 2.1.1.

Environment variables

O2HOME Is mandatory and contains the path to the O$_2$ installation directory.

O2SERVEROPTIONS See 2.1.1.

Example

The command below backs up system **sys** to device **/dev/mnt**. **o2dba_backup** stops when the backup finishes. It runs interactively. If volume change is needed, the user will be queried.

```
o2dba_backup -system sys -tape /dev/mnt
```

The following command backs up the system to device **/dev/mnt**. After the backup is completed, **o2dba_backup** continues to back up the cold log file until October 20 midnight of the current year.

```
o2dba_backup -tape /dev/mnt -witharchive -nointeractive
-date 10200000
```

The following command ensures a weekly backup. A backup session begins every Monday at 12 noon till next Monday before being succeeded by another session.

```
o2dba_backup -tape /dev/mnt -witharchive -nointeractive
-period 01-1200W
```

The following command resumes an interrupted backup session. Such options may be utilized after a severe system crash.

```
o2dba_backup -tape /dev/mnt -continue -nointeractive
```

The following is an example of an interactive session.

```
warning: Reach end of tape. Insert the next tape (#2) and
strike Enter ...

warning: Bad medium Id... Insert another tape and strike
Enter ...
```

FILES

Entries are generated in the backup history file during the backup session. The backup history file is found in the same directory as the system's catalog file. This is a per-system plain text file. Its name is composed by the system name and prefix **o2backuplog**.

SEE ALSO **o2dba_restore**, **o2dba_backup_display**, **o2dba_backup_label**

o2dba_backup_display

Summary

Display backup history

Syntax

o2dba_backup_display{**-file** *filename* |**-tape** *device_name*}
[**-system** *system_name*] [**-label** *string*]
[**-history** *filename*] [**-date** [*yy-*]*mmdd*[*HHMM*]] [**-verbose**]

o2dba_backup_display -version

o2dba_backup_display -help

o2dba_backup_display -env

Description

o2dba_backup_display analyzes the backup history and display the information in an user friendly way. The information provides:

- the period the system has been backed up;
- if a backup spans multi volumes, the identification of each volume;
- system states labeled during backup sessions;

o2dba_backup_display may be used to analyze the history file of a system backup in order to work out a plan for system restoration. **o2dba_backup_display** takes the default system's backup history file unless an argument *filename* is given. The file *filename* must be a backup history file.

Instead of reading backup history information from a backup history file, **o2dba_backup_display** can read information directly from system backup archives. The options **-file** or **-tape** are provided for this purpose.

Options

-date	Provides the events for the specified *date*.
-file	The name of the file which will contain the backup.
-history	Specifies the name of the historic file which must be used.

-label Associates the descriptive label *string* to the current system state.

-system Specifies the O_2 system name. If this option is not used, the system name given by the environment is used (see Default options below).

-tape The name of the device which will contain the backup.

-verbose Returns additional information about the operation.

-version Displays information about the current version and exits.

-help Displays information about the possible options and exits.

-env Displays the actual value taken for the options and exits. Allows you to verify that the option mechanism (explicit and implicit) is correctly set.

Default Options The previous o2 server options as well as other options can be implicitly set as explained in 2.1.1.

Environment variables

O2HOME Is mandatory and contains the path to the O_2 installation directory.

O2SERVEROPTIONS See 2.1.1.

Example

An example of a backup history file for a system may look like this:

```
Begin backup session system_name_22091997150154 in
/home/user/system_name/tmp/system_name at 22/09/1997 15:01
Beginning of media volume system_name_22091997150154_1 at
22/09/1997 15:01
State 22091997150101 label sep97 at 22/09/1997 15:01
Writing volume CatalVol -
/u/system_name/dve/home/bases/catalog_system_name at
22/09/1997 15:01
Done volume CatalVol -
/u/system_name/dve/home/bases/catalog_system_name at
22/09/1997 15:02
```

```
Writing volume DefaultVol -
/u/system_name/dve/home/bases/DefaultVol_3_system_name at
22/09/1997 15:02
Done volume DefaultVol -
/u/system_name/dve/home/bases/DefaultVol_3_system_name at
22/09/1997 15:02
Writing log
/u/system_name/dve/home/bases/coldlog_6_system_name
(Interval : 22/09/1997 15:01 - 22/09/1997 15:01) at
22/09/1997 15:02
Done log
/u/system_name/dve/home/bases/coldlog_6_system_name
(Interval : 22/09/1997 15:01 - 22/09/1997 15:01) at
22/09/1997 15:02
End complete system backup system_name_22091997150154 at
22/09/1997 15:02
State 22091997150201 label sep97 at 22/09/1997 15:02
End backup session system_name_22091997150154 in
/home/user/system_name/tmp/system_name at 22/09/1997 15:02
```

And the **o2dba_backup_display** reports something like this:

```
Backup session ‘sys_22091997150154' :
from: ‘22/09/1997 15:01:00' to: ‘22/09/1997 15:02:00'
backup media volumes:
volume: ‘sys_22091997150154_1'
states:
state ‘22091997150101' labeled ‘Avant_repas' at ‘22/09/1997
15:01:00'
state ‘22091997150201' labeled ‘Apres_repas' at ‘22/09/1997
15:02:00'
volumes list:
‘/u/sys/dve/home/bases/catalog_sys'
‘/u/sys/dve/home/bases/DefaultVol_3_sys'
log list:
‘/u/sys/dve/home/bases/coldlog_0_sys'
```

2 O2 System Commands

o2dba_backup_label

Summary Marks the current system state with a label

Syntax

```
o2dba_backup_label [-system system_name]
                      [-label string ] [-verbose]

o2dba_backup_label -version

o2dba_backup_label -help

o2dba_backup_label -env
```

Description **o2dba_backup_label** allows to explicitly give a new *label* to a system.

It is possible for the user to associate a system state with a text-formed description which is meaningful to the user. This is intended to be used by the administrator to make a landmark which is meaningful from the application's point of view.

A system state labeling entry is generated in the backup history log. The entry indicates the time stamp of the operation and the associated descriptive state label if it exists.

During restoration, the system administrator can set the time stamp associated with a particular label, using the option **date**. The list of labels can be displayed with the command **o2dba_backup_display.**

Options

-label Associates *string* as a descriptive label to the current system state.

-server Run the **o2server** process on *server_name*, if not already running.

-system Specifies the O_2 system name. If this option is not used, the system name given by the environment is used (see Default options below).

-verbose Returns additional information about the operation.

-version Displays information about the current version and exits.

-help Displays information about the possible options and exits.

-env Displays the actual value taken for the options and exits. Allows you to verify that the option mechanism (explicit and implicit) is correctly set.

Default Options The previous o2 server options as well as other options can be implicitly set as explained in 2.1.1.

Environment variables

O2HOME Is mandatory and contains the path to the O_2 installation directory.

O2SERVEROPTIONS See 2.1.1.

File Entries are generated in the backup history file during backup. By default, the backup history file is found in the same directory as the system's catalog file. This is a system specific plain text file. Its name is composed of the system name and suffix **o2backuplog**.

SEE ALSO **o2dba_restore, o2dba_backup, o2dba_backup_display**

o2dba_base_dump

Summary Builds a dump containing one or more bases from a given O_2 system.

Syntax

```
o2dba_base_dump {-file filename | -tape device_name}
                [-system system_name] [-server machine_name]
                [[-base base_name] ...] [-all] [-noindex]
                [-nocluster] [-verbose] [-optimize]
                [-name user_label]
                [-compressed]

o2dba_base_dump -version

o2dba_base_dump -help

o2dba_base_dump -env
```

Description **o2dba_base_dump** dumps the contents of one or more bases of an O_2 system to either a file or tape device.

You can enter the bases in the command or choose them from a list. If you enter them with the command, you choose external base names with the **-base** or **-all** options.

If you do not use either of the above options the system displays the list of all bases and you select the bases you want.

Starting from this list of bases, **o2dba_base_dump** detects all import-export relationships.

If the objects in one base reference objects in another base that is not selected, you can either:

a. restart **o2dba_base_dump** and choose all the bases concerned, in which case the inter-base relationship is preserved

b. or do nothing, in which case references to objects in the non-selected base are set to nil and a message is displayed.

o2dba_base_dump establishes a connection with an O_2 server running on the target system.

Options

-file	The name of a standard file which will contain the dump.
-compressed	The output dump is generated in a compressed format.
-tape	The name of a tape device which will contain the dump.

-name Write at the beginning of the dump a *user_label*.

-system Specifies the O_2 named system to connect to. An **o2server** must already be running on this system before **o2dba_base_dump** is invoked.

-server Specifies the machine name where **o2server** is running.

-base Specifies the external name of a base (optional). This option can be used repeatedly to provide a list of bases. If this option or the **-all** option is not used, a full list of bases is displayed for you to choose from.

-all All the bases in the given system are selected.

-noindex Do not dump index information. In this case **o2dba_base_load** can not restore indexes.

-nocluster Do not dump cluster information. Data is dumped but clustering information is not taken into account. In this case **o2dba_base_load** can not restore clustering.

-optimize Uses clustering information to optimize the physical position of data dumped.

-verbose Prints informative messages during the **o2dba_base_dump** session.

-version Displays information about the current version and exits.

-help Displays information about the possible options and exits.

-env Displays the current values for the various options and exits. Allows you to verify that the option mechanism (explicit and implicit) is correctly set.

Default Options The previous o2options and others can be implicitly set as explained in 2.1.2.

Environment variables

O2HOME Is mandatory and contains the path to the O_2 installation directory.

O2OPTIONS See 2.1.2.

Example

```
 o2dba_base_dump -system sys -server svr -base b1 -base b2
-file bases.dump
 o2dba_base_dump -system sys -server svr -all -tape
/dev/rmt0
```

The first example dumps the contents of bases **b1** and **b2** of the O_2 system **sys** into a standard file named **bases.dump**.

The second example dumps all bases of the O_2 system **sys** to a tape device **/dev/rmt0**.

o2dba_base_load

Summary

Loads data, previously dumped using **o2dba_base_dump**, into existing O_2 bases.

Syntax

```
o2dba_base_load {-file filename | -tape device_name}
                [-system system_name] [-server machine_name]
                [-noindex] [-nocluster] [-verbose] [-Force]

o2dba_base_load  {-file filename | -tape device_name}
                -identify

o2dba_base_load -version

o2dba_base_load -help

o2dba_base_load -env
```

Description

o2dba_base_load extracts the set of bases contained in a dump produced by **o2dba_base_dump** and updates the corresponding O_2 bases in the system.

Before starting this operation, ensure that:

a. the volumes required by the new data actually exist.

b. the schemas which contain the bases are compiled and are exactly the same as those in effect when the data was saved.

c. bases with the same names exist and are empty, that is, they contain no data, no clusters and no indexes.

You must not create indexes, clusters or base extensions, as they are restored to their original state when loading a base.

However, you can override the clusters and indexes, using the **-nocluster** and **-noindex** options.

o2dba_base_load establishes a connection with an O_2 server running on the target system.

Options

-file	The name of a standard file which will contain the dump.
-tape	The name of a tape device which will contain the dump.

-system Specifies the O_2 named system to connect to. An **o2server** must already be running on this system before **o2dba_base_load** is invoked.

-server Specifies the machine name where **o2server** is running.

-identify Display identifying information about the dumped base.

-noindex Ignore index definitions contained in the dump.

-nocluster Ignore clustering information contained in the dump.

-Force Force the restoration of a base without consistency check. This may render a database unusable! Refer to the *O_2 System Administration Guide*, in Schema Updates (base dump and base load).

-verbose Prints informative messages during the **o2dba_base_load** session.

-version Displays information about the current version and exits.

-help Displays information about the possible options and exits.

-env Displays the current values for the various options and exits. Allows you to verify that the option mechanism (explicit and implicit) is correctly set.

Default Options The previous o2options and others can be implicitly set as explained in 2.1.2.

Environment variables

O2HOME Is mandatory and contains the path to the O_2 installation directory.

O2OPTIONS See 2.1.2.

Example

```
o2dba_base_load -system sys -server svr -file bases.dump
o2dba_base_load -system sys -server svr -tape /dev/rmt0
```

The above examples load data previously dumped by using **o2dba_base_dump**. In the first example, the dumped data was stored in a standard file **bases.dump**, and in the second one the dumped data was stored in a tape whose device name is **/dev/rmt0**.

2 O2 System Commands

o2dba_check

Summary

Checks the consistency of all objects of a schema or base. You can check a closed schema, a transitive closure of schemas and bases linked by an import relationship or a base and its associated schema.

Syntax

o2dba_check [**-system** *system_name*] [**-server** *machine_name*] [[**-schema** *schema_name*] ...] [**-all_schemas**] [[**-base** *base_name*] ...] [**-all_bases**] [**-all**] [**-log** *logging_file*] [**-recover**] [**-verbose**] [**-help**] [**-Validate** *number_of_objects*] [**-CheckingLevel {0|1}**]

o2dba_check -version

o2dba_check -help

o2dba_check -env

Description

o2dba_check generates information about erroneous references among objects and complex values in an O_2 system. If **o2dba_check** is run in recover mode, erroneous references are replaced as follows:

- references to objects are replaced by a nil pointer.
- references to complex values are replaced by a corresponding value that is initialized with O_2 default values.

o2dba_check operates on a list of schemas and bases stored in the system, and displays information directly or returns it to a file. You enter the schemas and bases in the command or you can choose them from a list.

Options are available to select bases and schemas by their external names.

There are also group options such as **all**, **all_schemas** and **all_bases**.

If you do not use one of these options the list of system schemas and bases is displayed and you must choose from this.

Once schemas and bases have been selected **o2dba_check** discovers schemas and bases linked through import relationships to those selected.

o2dba_check establishes a connection with an O_2 server which must already be running on the system.

The **o2dba_check** operation can be performed in multi-user mode.

o2dba_check can perform the following two types of consistency checks:

- verification of referential integrity (there are no dangling inter-object references).
- verification of the composition graph as per the corresponding schema description.

By default (i.e., without the option **CheckingLevel** or **CheckingLevel 0**), both these types of consistency checks are carried out in depth.

If you use the option **CheckingLevel 1**, the first type of consistency check is carried out completely; for the second type of consistency check, only the first level of type comparison is carried out.

The first level of type comparison implies that a reference that is supposed to point to a tuple does not point to a collection (and vice versa). If the tuple object that is pointed to is not an instance of the expected class, or if the expected collection is a set of integers and the real collection is a bag of Persons, this inconsistency is not reported.

Options

-system Specifies the O_2 named system to connect to. An **o2server** must already be running on this system before **o2dba_check** is invoked.

-server Specifies the machine name where **o2server** is running.

-schema Specifies the external name of a schema on the system. This option can be repeated to enter several schemas.

-base Specifies the external base name belonging to the system. This option can be repeated to give a list of bases to **o2dba_check**.

-all_schemas Selects all schemas of a system.

-all_bases Selects all bases of a system.

-all Selects all schemas and bases of a system.

-log Returns information on erroneous references to the file named *logging_file*. If no directory is specified, the current directory is used to store the log file.

-recover Activates recovery. Erroneous references to objects are replaced by nil.

-verbose Returns additional information about the operation.

-help Displays information about the possible options and exits.

-version Displays information about the current version and exits.

-env Displays the actual value taken for the options and exits. Allows you to verify that the option mechanism (explicit and implicit) is correctly set.

-Validate Allows you to verify a base or a schema in more than one transaction. A new transaction starts each time *number_of_objects* objects are processed. To ensure that the vue of the base or the schema is coherent, you must use this option with a server in single-user mode. This option is particularly useful when the running of **o2check** on a base causes many object updates due to schema modification.

-CheckingLevel Allows you to set the level of consistency checks.

Default Options The previous o2options and others can be implicitly set as explained in 2.1.2.

Environment variables

O2HOME Is mandatory and contains the path to the O_2 installation directory.

O2OPTIONS See 2.1.2.

o2dba_copy

Summary

Copies an O_2 named database system to another

Syntax

o2dba_copy **-source** *system_name1* **-target** *system_name2*
[**-force**] [**-server** *machine_name*] [**-verbose**]
[**-volumes** *filename*]

o2dba_copy -version

o2dba_copy -help

o2dba_copy -env

Description

The **o2dba_copy** program copies the contents of one named database system to another.

The name of the source system is specified with the mandatory **-source** argument, and that of the target system with the mandatory **-target** argument.

Both system names must be defined in the configuration file .o2serverrc in the O_2 installation directory. The **o2dba_copy** program first checks that enough disk space exists for the copy. If there is not enough space, no copy is performed and a message is displayed.

Otherwise, it removes any existing system with the same name as the target system (*system_name2*), asking for confirmation first unless the **-force** option is specified; it then copies the catalogue volume of *system_name1* into the catalogue volume of *system_name2*.

Finally, each of the user volumes of *system_name1* is copied to *system_name2*. The program asks you for the target location of each user volume (unless the **-volumes** option is specified).

The **o2dba_copy** program invokes the **o2server** program for its own purposes. Any existing **o2server** processes running on the same system must be terminated to avoid conflict. If this has not been done, **o2dba_copy** informs you and halts.

Options

-force Overwrites any existing target system without seeking confirmation.

-server Run the **o2server** process for the source and the target systems on *machine_name*.

-volumes Rather than asking for user volume locations, take the relevant information from file *filename*. The contents of this file must be a list of volume path names with one identifier and path name per line corresponding to the locations into which the user volumes of the target system are to be copied. The file should thus contain one line per user volume in the source system. See the example of the syntax above. The **-volumes** option is most useful when invoking **o2dba_copy** in shell scripts.

-verbose Returns additional information about the operation.

-version Displays information about the current version and exits.

-help Displays information about the possible options and exits.

-env Displays the actual value taken for the options and exits. Allows you to verify that the option mechanism (explicit and implicit) is correctly set.

All the options follow the precedence rules described in 2.1.1.

Environment variables

O2HOME Is mandatory and contains the path to the O_2 installation directory.

O2SERVEROPTIONS See 2.1.1.

Example of the -volumes optionfile

```
create volume vol1 in /newpath/bases/vol1
create volume vol2 in /newpath/bases/vol2
extend volume vol2 in /newpath/bases/vol2_ext1
```

o2dba_env

Summary

Displays environment details for an O_2 database.

Syntax

o2dba_env [**-system** *system_name*] [**-server** *machine_name*] [**-verbose**]

o2dba_env -version

o2dba_env -help

o2dba_env -env

Description

The **o2dba_env** program reports on the environment and the physical volumes of an O_2 database. The report shows the internal environment for the system, including the names and locations of O_2 executable programs and resource files. The catalogue, shadow, log and all user volumes associated with the named system are listed along with their sizes. The report is based on the current contents of the catalogue volume of the named system. The **o2dba_env** program invokes the **o2server** program for its own purposes or uses an existing **o2server** if one is running.

Options

-system Specifies the system name to be reported. If this option is not used, the system name is taken from the default options below.

-server Run the **o2server** process on *machine_name*, which must be the name of a machine on the network.

-verbose Prints informative messages during the **o2dba_env** session.

-version Displays information about the current version and exits.

-help Displays information about the possible options and exits.

-env Displays the actual value taken for the options and exits. Allows you to verify that the option mechanism (explicit and implicit) is correctly set.

Default Options

The previous o2server options and others can be implicitly set as explained in 2.1.1.

Environment variables

`O2HOME` Is mandatory and contains the path to the O_2 installation directory.

`O2SERVEROPTIONS` See 2.1.1.

o2dba_init

Summary

Initializes an O_2 database system

Syntax

```
o2dba_init [-system system_name] [-server machine_name]
           [-catalog catalogue] [-force] [-verbose]

o2dba_init -version

o2dba_init -help

o2dba_init -env
```

Description

The **o2dba_init** program formats and initializes the contents of an O_2 database system.

Any existing O_2 system with the same name is removed, without prompting if the **-force** option is used.

The catalogue volume for the specified system name is taken from the O2HOME installation directory or from the specified directory.

By default, this base catalogue is the file **catalog_o2ref** found in the **bases** subdirectory of the O_2 installation directory.

All of the system volume specifications must be present in the .o2serverrc configuration file in the O_2 installation directory, prior to running **o2dba_init**.

In addition, **o2dba_init** creates one user volume for the named system and populates it with system-supplied objects, methods and functions.

The user volume is created in the same directory as the catalogue volume, and has the name **DefaultVol**.

The **o2dba_init** program invokes the **o2server** program for its own purposes. Any existing **o2server** processes running on the same system must be terminated to avoid protocol conflicts. If this has not been done, **o2dba_init** informs you and halts.

Options

-system Specifies the O_2 system name to initialize. If this option is not used, the system name is given by the environment used (see Default options below).

-server Run the **o2server** process on *machine_name*, which must be the name of a machine on the network.

-catalog Specifies the full path and filename of a base catalogue to use in place of the standard one in the **bases** subdirectory.

-force No confirmation required to remove a system with the same name.

-verbose Returns additional information about the operation.

-version Displays information about the current version and exits.

-help Displays information about the possible options and exits.

-env Displays the actual value taken for the options and exits. Allows you to verify that the option mechanism (explicit and implicit) is correctly set.

Default Options The previous o2 server options and others can be implicitly set as explained in 2.1.1.

Environment variables

O2HOME Is mandatory and contains the path to the O_2 installation directory.

O2SERVEROPTIONS See 2.1.1.

o2dba_monitor

Summary

Monitor O_2 transactions and server sessions.

Syntax

```
o2dba_monitor [-system system_name] [-server server_name]
              [-list] [-kill {xid | all}]
              [-shutdown now | nowarn | minutes]
              [-log on | off | close]
              [-recover log_file_name[-force]] [-stat] [-reset]
              [-licence]
```

```
o2dba_monitor -version
```

```
o2dba_monitor -help
```

```
o2dba_monitor -env
```

Description

The **o2dba_monitor** command calls up a utility to list or abort transactions, collect statistics on transactions or shutdown a system.

It is also used to manage the cold log for recovery purposes.

If you use none of the options given below, **o2dba_monitor** indicates if and where a server is currently running on that system.

If the server is in the process of shutting down **o2dba_monitor** cannot connect to it.

Options

-system Specifies the O_2 system name. If this option is not used, the system name is given by the environment used (see Default options below).

-server Specifies the machine name where **o2server** is running.

-list This command lists active transactions.

-kill This kills a transaction identified by *xid* or all the transactions currently running if you use the **all** option.

-shutdown Allows you to either immediately stop the server using the **now** option, stop the server without broadcasting a shutdown message using the **nowarn** option, or shut down after a specified time using the *minutes* option which has a default value of one minute.

If no clients are running the minutes value is ignored and the shutdown occurs immediately.

If **nowarn** is not used and if at least one client is active a broadcast message is sent to each window of the workstation running the client(s), indicating that the server running on a given system will be stopped immediately or in a given number of minutes.

-log Enables you to turn **on**, **off** or to **close** the server cold logging. It is better to use the .o2serverrc configuration file to enable logging rather than the option **log on**.

If you use **log on** the server creates a file called **coldlog_***system_name* and starts logging all the modifications carried out by committed transactions.

If you use **log off** the server closes the **coldlog_***system_name* and renames it **coldlog_***n_system_name* where n is a number which increases, starting at 0. Cold logging stops.

If you use **log close**, this has the same effect as log off except that it is immediately followed by log on. It guarantees that no transaction can commit (i.e. there is no information loss) during the switch. This option is used to save the generated logs periodically.

When the **coldlog_***system_name* file is closed O_2 renames it as **coldlog_***sequence_number_system_name*. The sequence number is increased each time the file is closed. These files can then be stored and reused in their order of sequence to recover the system.

-recover Enables you to apply all logged modifications to a backed up system. The first log of the list is the log which was started just after the system backup. The others must be given in their order of sequence.

If the **force** option is set, the program restores the volumes in the directories specified in the .o2serverrc file. Otherwise the user is asked for the location of the volumes.

o2server is automatically started when you use the **recover** option. Do not use **o2server** to recover a system.

-stat Display server statistics.

-reset Resets all server statistics counters to 0.

For further details on statistics refer to the *O_2 System Administration Guide*.

-verbose Returns additional information about the operation.

-version Displays information about the current version and exits.

-help Displays information about the possible options and exits.

-env Displays the actual value taken for the options and exits. Allows you to verify that the option mechanism (explicit and implicit) is correctly set.

-licence Displays the status of your licence.

Default Options The previous o2 server options and others can be implicitly set as explained in 2.1.1.

Environment variables

O2HOME Is mandatory and contains the path to the O_2 installation directory.

O2SERVEROPTIONS See 2.1.1.

o2dba_move_volume

Summary Moves O_2 volumes to other operating system files.

Syntax

```
o2dba_move_volume [-system system_name] [-server machine_name]
                  -source old_path -target new_path
                  [-verbose]

o2dba_move_volume -version

o2dba_move_volume -help

o2dba_move_volume -env
```

Description The **o2dba_move_volume** program changes the physical location of an O_2 volume or extension. The complete path of your O_2 volume to move is specified with the mandatory **-source** argument, and the new location is specified by the mandatory **-target** argument. This program moves the contents of the O_2 volume into a file specified by the new complete path name of the volume.

The **o2dba_move_volume** program invokes the **o2server** program for its own purposes. Any existing **o2server** processes running on the same system must be terminated to avoid protocol conflicts. If this has not been done, **o2dba_move_volume** informs you and halts.

Options

-system Specifies the O_2 system name. If this option is not used, the system name given by the environment variable is used (see Default options below).

-server Run the **o2server** process for the source and the target systems on *machine_name*.

-source The path to the volume that you want to move (an OS file where the volume resides).

-target The path to the file to which you want to move your volume.

-verbose Returns additional information on the operation.

-version Displays information about the current version and exits.

-help Displays information about the possible options and exits.

-env Displays the actual value taken for the options and exits. Allows you to verify that the option mechanism (explicit and implicit) is correctly set.

Default Options The previous o2 server options and others can be implicitly set as explained in 2.1.1.

Environment variables

O2HOME Is mandatory and contains the path to the O_2 installation directory.

O2SERVEROPTIONS See 2.1.1.

o2dba_restore

Summary

Restores a system from its backup

Syntax

o2dba_restore {**-file** *filename* | **-tape** *device_name*}
[**-system** *system_name*] [**-server** *server_name*]
[**-id** *identify*][**-recover** *cold_log_filename*][**-identity**]
[**-volumes** *vol_loc_filename*] [**-verbose**] [**-date**
[*yy-*]*mmdd*[*HHMM*]]

o2dba_restore -version

o2dba_restore -help

o2dba_restore -env

Description

o2dba_restore restores a system from its backup archives obtained by **o2dba_backup**. The system *system_name* must not exist.

After being launched, **o2dba_restore** restores a system by reading backup media volumes in the same order as they were written.

If the backup has been obtained without option **-witharchive**, the **o2dba_restore** stops after the system is restored.

If the backup was created with option **-witharchive**, it contains a full system backup with cold log archives. **o2dba_restore** restores the system and the cold log archives. If no target *date* is specified, **o2dba_restore** keeps reading archives as long as there are backup archives left. If a target *date* is specified, **o2dba_restore** stops as soon as the system is restored to that date.

If a backup was created with option **-archivelog**, it contains only cold log archives. **o2dba_restore** updates the system with these log files. It is the user's responsibility to restore the system to the state just before the log files were archived before using **o2dba_restore** to update the system.

o2dba_restore may require user intervention. If a backup spans multi-volumes, **o2dba_restore** will informed the user when to carry out volume changes. **o2dba_restore** then verifies that the mounted backup media volume is operational and part of the backup. If this is not the case **o2dba_restore** notifies the user and waits for the volume to be replaced.

There exist multiple scenarios for restoring an O_2 system after a disk crash.

1. You know precisely which backup must be restored. In this case, place the first volume of the backup in the tape reader (option **-tape**) or enter the name of the file which contains the first volume (option **-file**), and the tool restores the O_2 system from the backup. As an extra precaution, you can specify the identifier of the backup, using option **-id**. In this case, **o2dba_restore** verifies that this identifier corresponds to the identifier found in the backup. If the two identifiers do not match, an error message is displayed and **o2dba_restore** halts.

2. You do not know precisely which backup tape or file must be restored, but know to which date you would like to restore the O_2 system. In this case, use **o2dba_backup_display** to locate the first volume of the backup (see **o2dba_backup_display**). Then place the first volume of the backup in the tape reader (option **-tape**) or enter the name of the file which contains the first volume (option **-file**), and the tool restores the O_2 system from the backup. You should provide the date until which the restoration should be carried out (option **-date**). In this mode, the tool restores the O_2 system from the backup to the date specified with option **-date** .

3. You do not know precisely which backup tape or file must be restored, nor which date corresponds to the desired O_2 system, but you know the label corresponding to the desired O_2 system. In this case, use **o2dba_backup_display** to locate the first volume of the backup and the date associated with this label (see **o2dba_backup_display**). Then place the first volume of the backup in the tape reader (option **-tape**) or enters the name of the file which contains the first volume (option **-file**), and the tool restores the O_2 system from the backup. You should provide the date to which the restoration should be carried out (option **-date**). In this mode, the tool restores the O_2 system from the backup until the specified date from the **-date** option. For cases 2 and 3, as an extra precaution, you can specify the identifier of the backup, using option **-id**, obtained with **o2dba_backup_display**.

In certain scenarios, the restoration is completed by restoring the cold log of the server (using option **-recover**). In effect, it can happen that the last backup does not correspond to the state of the system at the time of a disk crash. In such a situation, if you wish to restore the system to the state before the crash, you must first restore the most recent backup and then apply the cold logs of the server (option **-recover**). The first cold log to apply corresponds to the cold log following the last cold log that was restored from the backup. Then apply the cold logs one after the other to complete all the cold logs up to the current one inclusive.

Options

-date [yy-]mmdd[HHMM]
Specifies the date to which the system should be restored. The utility restores the system to the state just before this date.

H	hour - 00 to 23
M	minute - 00 to 59
d	day of month - 01 to 31
m	month of year - 01 to 12
y	year - 00 to 99

-id Specifies the backup identifier to be used. **o2dba_restore** verifies the version contained in the backup media corresponds to the version being asked. Backup identifier is logged in the backup history file. By using **o2dba_backup_display** we obtain informations about backup sessions and their identifiers.

-identify Dumps backup information from backup media. This needs the first volume of the backup media to be read. The information includes backup identifier, time stamp of the backup etc.

-file The name of the file which contains the backup.

-recover Recovers the system from file *filename*. The file must be a cold log file. This option is used to access a cold log file that has not been backed up before the system crash, to recover transactions logged in the file. This file must be recovered after the archived log files.

-server Run the **o2server** process on *server_name*, if not already running.

-system Specifies the O_2 system name. If this option is not used, the system name given by the environment is used (see Default options below).

-tape The name of the device which contains the backup.

-volumes Provides the system-required volume locations in a file *vol_loc_filename*, rather than providing this information interactively. The contents of this file must be a list of standard O_2 commands used to create new volumes or extend existing volumes. The file should contain one line per user volume of the backup file/ tape, e.g.:

```
create volume VolName1 in /u/tmp

extend volume VolName2 in /u/tmp2
```

The **-volumes** option is useful when invoking **o2dba_restore** in shell scripts.

-verbose Returns additional information about the operation.

-version Displays information about the current version and exits.

-help Displays information about the possible options and exits.

-env Displays the actual value taken for the options and exits. Allows you to verify that the option mechanism (explicit and implicit) is correctly set.

Default Options

The previous o2 server options as well as other options can be implicitly set as explained in 2.1.1.

Environment variables

O2HOME Is mandatory and contains the path to the O_2 installation directory.

O2SERVEROPTIONS See 2.1.1.

Example

Example of the -volumes optionfile

```
create volume vol1 in /newpath/bases/vol1
create volume vol2 in /newpath/bases/vol2
extend volume vol2 in /newpath/bases/vol2_ext1
```

The command below restores system **sys** from backup device **/dev/mnt**. It runs interactively. If volume change is needed, the user will be queried.

```
o2dba_restore -system sys -tape /dev/mnt
```

The following command restores the system from device **/dev/mnt** to the state it was on October 20, 1997. The backup history file of the system which had been saved is **~/bkuphist**.

```
o2dba_restore -tape /dev/mnt -date 97-1020
```

Suppose after a device failure, we have a system backup which contains up to the last cold log file generated by the system and we have succeeded to save the current log file of the system after the crash. After having restored the system from its backup, the following command could be used to update the system to the state just before the crash.

```
o2dba_restore -system sys -recover ~/saved_log
```

SEE ALSO **o2dba_backup**, **o2dba_backup_label**, **o2dba_backup_display**

o2dba_rm

Summary Removes an O_2 named database system

Syntax

o2dba_rm [**-system** *system_name*] [**-server** *machine_name*] [**-force**] [**-verbose**]

o2dba_rm -version

o2dba_rm -help

o2dba_rm -env

Description The **o2dba_rm** program deletes all physical volumes of a named database system. The catalogue, shadow and log volumes as well as all user volumes associated with the named system are deleted. The **o2dba_rm** program invokes the **o2server** program for its own purposes. Any existing **o2server** processes running on the same system must be terminated to avoid protocol conflicts. If this has not been done, **o2dba_rm** warns the user and halts.

Options

-system Specifies the O_2 system name to remove. If this option is not used, the system name given by the environment is used (see Default options below).

-server Runs the **o2server** process on *machine_name*, which must be the name of a machine on the network.

-force No confirmation required to remove the system.

-verbose Returns additional information about the operation.

-version Displays information about the current version and exits.

-help Displays information about the possible options and exits.

-env Displays the actual value taken for the options and exits. Allows you to verify that the option mechanism (explicit and implicit) is correctly set.

Default Options The previous o2 server options and others can be implicitly set as explained in 2.1.1.

Environment variables

O2HOME	Is mandatory and contains the path to the O_2 installation directory.
O2SERVEROPTIONS	See 2.1.1.

o2dba_schema_dump

Summary

Builds a dump containing one or several schema definitions.

This command dumps the definition of one or several O_2 schemas into a file or tape device.

Syntax

```
o2dba_schema_dump {-file filename | -tape device_name}
                  [-system system_name] [-server machine_name]
                  [[-schema schema_name] ...|-all] [-sources]
                  [-name user_label] [-verbose]

o2dba_schema_dump -version

o2dba_schema_dump -help

o2dba_schema_dump -env
```

Description

o2dba_schema_dump is used to construct a dump of one or several schemas from an O_2 system.

You specify the names of the schemas to be dumped using either the **-schema** or the **-all** options or by selecting them from a list displayed by **o2dba_schema_dump**.

This dump is intended to be used by **o2dba_schema_load** to load the schema onto the target system.

For each schema name, **o2dba_schema_dump** establishes a list of complementary schemas and bases imported by the selected schemas. Only the schemas selected by the user are copied to the dump. The contents of the imported schemas and bases, if any, are not copied.

The imported schemas which are not stored in the dump must exist in the target system before carrying out the **o2dba_schema_load** operation.

The imported bases are created by the load operation if they do not already exist.

Refer to section 8.10 of the *O_2 System Administration Guide* for further explanations.

o2dba_schema_dump establishes a connection with an O_2 server running on the system specified.

The **o2dba_schema_dump** operation must be performed in single user mode only. This is done by setting the **-mode** option of the **o2server** command to **single-user**.

Options

-file The name of a standard file which will contain the dump.

-tape The name of a tape device which will contain the dump.

-name Write at the beginning of the dump a *user_label*.

-system Specifies the O_2 named system to connect to. An **o2server** must already be running on this system before **o2dba_schema_dump** is invoked.

-server Specifies the machine name where **o2server** is running.

-schema Specifies the external name of a schema to be dumped. This option can be used repeatedly in order to give a list of schemas to **o2dba_schema_dump**.

If the **all** or **schema** option is not used, you choose from a list of schemas displayed by **o2dba_schema_dump**.

-all Select all schemas on the same system.

If the **all** or **schema** option is not used, you choose from a list of schemas displayed by **o2dba_schema_dump**.

-sources The source code of the schema elements (classes, name etc.) is included in the dump if it exists.

-verbose Prints informative messages during the **o2dba_schema_dump** session.

-version Displays information about the current version and exits.

-help Displays information about the possible options and exits.

-env Displays the current values for the options and exits. Allows you to verify that the option mechanism (explicit and implicit) is correctly set.

Default Options The previous o2options and others can be implicitly set as explained in 2.1.2.

Environment variables

O2HOME Is mandatory and contains the path to the O_2 installation directory.

O2OPTIONS See 2.1.2.

Example

```
o2dba_schema_dump -system sys -server svr -schema sch1
-schema sch2  -file schema.dump
o2dba_schema_dump -system sys -server svr -all -tape
/dev/rmt0
```

The first example saves the contents of schema **sch1** and schema **sch2** into a standard file named **schema.dump**. The second example dumps all schemas of the O_2 system **sys** to a tape device **/dev/rmt0**.

o2dba_schema_load

Summary

Loads a set of schemas into an O_2 target system.

Syntax

```
o2dba_schema_load {-file filename | -tape device_name }
                  [-system system_name] [-server machine_name]
                  [-force] [-verbose]  [-sources]

o2dba_schema_load {-file filename |  -tape device_name}
                 -identify

o2dba_schema_load -version

o2dba_schema_load -help

o2dba_schema_load -env
```

Description

o2dba_schema_load extracts the set of schemas contained in a dump produced by **o2dba_schema_dump** and creates or updates the corresponding O_2 schemas in a target system.

o2dba_schema_load is invoked with a *filename* or *device_name*. Schemas are identified by their external names.

If a given schema name found in the dump exists in the *system_name*, it is replaced. An imported base found in dump is installed in *system_name* only if *system_name* does not contain a base with the same name. In this case, the base is created empty. The contents of imported bases are never saved by **o2dba_schema_dump**.

When you load a schema onto a target system where the schema already exists, **o2dba_schema_dump** checks that the two schema versions are compatible. If they are not compatible, the operation is aborted.

Refer to section 8.10 of the *O_2 System Administration Guide* for further explanations concerning consistency checks.

o2dba_schema_load establishes a connection with an O_2 server running on the target system. The operation must be performed in single user mode only. This is done by setting the **-mode** option of the **o2server** command to **single-user**.

Options

-file Name of the file which contains the dump.

-tape Name of the tape device which contains the dump.

-system Specifies the O_2 named system to connect to. An **o2server** must already be running on this system before **o2dba_schema_load** is invoked.

-server Specifies the machine name where **o2server** is running.

-sources Restores the source code if it exists in the dump for a schema. Note that a schema without source code can only be managed using the **o2api_shell**. It is however possible to recreate class source code by using the o2shell command **create sources**.

Warning: If you do not use this option, any schema source codes in the target system are overwritten by the schema without source code.

-identify Displays identifying information about the dump file or device.

-force No confirmation required before each schema load.

-verbose Prints informative messages during the **o2dba_schema_load** session.

-version Displays information about the current version and exits.

-help Displays information about the possible options and exits.

-env Displays the current values for the various options and exits. Allows you to verify that the option mechanism (explicit and implicit) is correctly set.

Default Options The previous o2options as well as other options can be implicitly set as explained in 2.1.2.

Environment variables

O2HOME Is mandatory and contains the path to the O_2 installation directory.

O2OPTIONS See 2.1.2.

Example

```
o2dba_schema_load -system sys -server svr -file schema.dump
o2dba_schema_load -system sys -server svr -tape /dev/rmt0
```

The above examples load schemas previously dumped by using **o2dba_schema_dump**. In the first example, the schema was dumped into a standard file **schema.dump**. In the second example, the dumped schema are on the tape device **/dev/rmt0**.

o2dba_shell

Summary

Calls up the O_2 database administration utility. This gives you access to O_2DBA commands.

Syntax

o2dba_shell [**-system** *system_name*] [**-server** *machine_name*] [**-verbose**] [**-alpha**]

o2dba_shell -version

o2dba_shell -help

o2dba_shell -env

Description

You must enter the **o2dba_shell** command to use database administration commands and OQL query sessions. **o2dba_shell** makes available those commands which do not require a development license. These commands are described in Chapter 3.

o2dba_shell establishes a connection with a named O_2 database system using **o2server**, which must already be running.

Options

-system Specifies the O_2 named system to connect to. An **o2server** must already be running on this system before **o2dba_shell** is invoked.

-server Specifies the machine name where **o2server** is running.

-alpha Disables the graphical interface for database browsing. Uses a dumb terminal style.

-verbose Prints informative messages during the **o2dba_shell** session.

-version Displays information about the current version and exits.

-help Displays information about the possible options and exits.

-env Displays the actual value taken for the options and exits. Allows you to verify that the option mechanism (explicit and implicit) is correctly set.

Default Options

The previous o2options and others can be implicitly set as explained in 2.1.2.

Environment variables

O2HOME Is mandatory and contains the path to the O_2 installation directory.

O2OPTIONS See 2.1.2.

o2dba_shutdown

Summary

Stops a server

Syntax

```
o2dba_shutdown [-system system_name] [-server machine_name]
               {[-now] | [-nowarn] | [-time<minutes>]}
               [-verbose]
```

```
o2dba_shutdown -version
```

```
o2dba_shutdown -help
```

```
o2dba_shutdown -env
```

Description

The **o2dba_shutdown** utility allows you to stop a given system properly.

Options

-system Specifies the O_2 system name to shutdown. If this option is not used, the system name given by the environment is used (see Default options below).

-server Specifies the machine where **o2server** runs.

-now Using this option stops the server immediately.

If the **-nowarn** option is also used the server is stopped immediately without broadcasting a shutdown message.

-nowarn The default warning message is not sent. A default warning message is sent to each client system. The message states that the server running on a given system will be stopped immediately or in a given number of minutes.

-time This option allows you to stop the server in a given number of minutes.

If this option is not used, a default of one minute is used. If no clients are running, shutdown is immediate.

-verbose Returns additional information about the operation.

-version Displays information about the current version and exits.

-help Displays information about the possible options and exits.

-env Displays the actual value taken for the options and exits. Allows you to verify that the option mechanism (explicit and implicit) is correctly set.

Default Options The previous o2 server options and others can be implicitly set as explained in 2.1.1.

Environment variables

O2HOME Is mandatory and contains the path to the O_2 installation directory.

O2SERVEROPTIONS See 2.1.1.

Example

```
o2dba_shutdown -system sys.
```

o2dsa_shell

Summary

Calls up the O_2 schema administration utility. This gives you access to O_2DSA commands. If you are using O_2C you do not need to use the **o2dsa_shell** command.

Syntax

o2dsa_shell [**-system** *system_name*] [**-server** *machine_name*] [**-verbose**] [**-alpha**]

o2dsa_shell -version

o2dsa_shell -help

o2dsa_shell -env

Description

You must enter the **o2dsa_shell** command to use schema administration commands and OQL query sessions. **o2dsa_shell** makes available those commands which require a development license. These commands are described in Chapter 4.

o2dsa_shell establishes a connection with a named O_2 database system using **o2server**, which must already be running.

Options

-system Specifies the O_2 named system to connect to. An **o2server** must already be running on this system before **o2dsa_shell** is invoked.

-server Specifies the machine name where **o2server** is running.

-alpha Disables the graphical interface for database browsing. Uses a dumb terminal style.

-verbose Prints informative messages during the **o2dsa_shell** session.

-version Displays information about the current version and exits.

-help Displays information about the possible options and exits.

-env Displays the actual value taken for the options and exits. Allows you to verify that the option mechanism (explicit and implicit) is correctly set.

Default Options

The previous o2options and others can be implicitly set as explained in 2.1.2.

Environment variables

O2HOME	Is mandatory and contains the path to the O_2 installation directory.
O2OPTIONS	See 2.1.2.

o2patch

Summary

Patches an O_2 database system.

Syntax

o2patch -system *system_name* [**-server** *machine_name*]

Description

o2patch is used to upgrade a version from one release to a more recent one.

For example, you use **o2patch** to upgrade from release 5.1.1 to 5.1.2.

To find out which version and/or release you have, type **o2dba_shell -version**.

For installation instructions refer to the installation sheet which is delivered with the version.

To start **o2patch**, an **o2server** processing the system called *system_name* must be running.

Options

-system The name of the O_2 system to patch. A system name must be specified with the **-system** option; otherwise **o2patch** cannot start.

-server Specifies the name of the machine where o2server is running.

o2server

Summary

Starts a server for an O_2 database session

Syntax

o2server [**-system** *system_name*] [**-verbose**][**-mode** *mode*]
[**-foreground**]

o2server -version

o2server -help

o2server -env

Description

The **o2server** program provides a connection between O_2 and the physical storage volumes associated with a logical, named database system.

An **o2server** process is started automatically by some O_2 programs (**o2dba_copy**, **o2dba_env**, **o2dba_init**, **o2dba_rm, o2dba_backup and o2dba_restore**), and is terminated by these programs as they exit.

For other programs, **o2server** must be invoked manually prior to starting an O_2 session. The **o2server** is started up in background unless you use the **foreground** option.

o2server must be terminated using **o2dba_shutdown**. If the server is shutting down an o2 client as **o2dba_monitor>** cannot connect to the o2server

There must be one and only one **o2server** process executing for each name system in use.

When the **o2server** process is started, it creates and formats the shadow volume with an elastic size. This volume is removed when **o2server** exits.

o2server also resizes the log volume if necessary when it starts. To change the log size, you change the .o2serverrc file and restart the server.

If **o2server** is interrupted by a hardware or power failure, for example, simply restart it. A warm recovery mechanism ensures data integrity. See the *O_2 System Administration Guide* for more details.

Options

-system Specifies the O_2 system name. If this option is not used, the system name given by the environment is used (see Default options below).

-verbose In verbose mode, **o2server** returns details of important events such as transactions, creation of O_2Store files, connection of users, etc.

-mode This option specifies the mode in which the server operates in terms of concurrency control, rollbacks and crash recovery. The following values are accepted:

multi-user Multi-user concurrency control enabled, Aborts enabled, Recovery enabled

single-user Multi-user concurrency control disabled, Aborts enabled, Recovery enabled

transaction-off Multi-user concurrency control disabled, Aborts disabled, Recovery disabled

-version Displays information about the current version and exits.

-help Displays information about the possible options and exits.

-env Displays the actual value taken for the options and exits. Allows you to verify that the option mechanism (explicit and implicit) is correctly set.

Default Options The previous o2 server options and others can be implicitly set as explained in 2.1.1.

Environment variables

O2HOME Is mandatory and contains the path to the O_2 installation directory.

O2SERVEROPTIONS See 2.1.1.

o2shell

Summary Starts an O_2 development database session.

Syntax

```
o2shell [-system system_name] [-server machine_name]
        [-alpha][-verbose]
        [-libs lib1:lib2...libn]
        [-libpath path1:path2...pathn]

o2shell -version

o2shell -help

o2shell -env
```

Description This command calls up an environment which allows you to develop and test O_2 applications using O_2C. O_2DBA and O_2DSA commands are also available via **o2shell**. See the *O_2C Reference Manual* for O_2C specific commands.

You need an O_2C development license to run **o2shell**.

o2shell establishes a connection with a named O_2 system for which an **o2server** must be active.

Options

-system Specifies the O_2 named system to connect to. An **o2server** must already be running on this system before **o2shell** is invoked.

-server Specifies the machine name where **o2server** is running.

-alpha Disables the graphical interface for database browsing. Uses a dumb terminal style.

-verbose Prints informative messages during the **o2shell** session.

-libs Causes the O_2C compiler to look for external functions in libraries named **lib***lib1***.xx**, where **xx** depends on the operating system. It must be a shared library. The libraries are sought first in directories specified by the **-libpath** option prior to the standard library directories.

-libpath Causes the O_2C compiler to look for libraries in the specified directories before searching in the standard library directories.

-version Displays information about the current version and exits.

-help Displays information about the possible options and exits.

-env Displays the actual value taken for the options and exits. Allows you to verify that the option mechanism (explicit and implicit) is correctly set.

Default Options The previous o2options and others can be implicitly set as explained in 2.1.2.

Environment variables

O2HOME Is mandatory and contains the path to the O_2 installation directory.

O2OPTIONS See 2.1.2.

o2tools

Summary

Starts a graphical O_2 development database session.

Syntax

```
o2tools [-system system_name] [-server machine_name]
        [-verbose]
        [-libs lib1:lib2...libn]
        [-libpath path1:path2...pathn]

o2tools -version

o2tools -help

o2tools -env
```

Description

This command calls up an environment which allows you to develop and test O_2 applications using the graphical development environment. See the *O_2Tools User Manual*.

You need an O_2Tools development license to run **o2tools**.

o2tools establishes a connection with a named O_2 system for which an **o2server** must be active.

Options

-system Specifies the O_2 named system to connect to. An **o2server** must already be running on this system before **o2tools** is invoked.

-server Specifies the machine name where **o2server** is running.

-verbose Prints informative messages during the **o2tools** session.

-libs Causes the O_2C compiler to look for external functions in libraries named **lib***lib1***.xx**, where **xx** depends on the operating system. It must be a shared library. The libraries are sought first in directories specified by the **-libpath** option prior to the standard library directories.

-libpath Causes the O_2C compiler to look for libraries in the specified directories before searching in the standard library directories.

-version Displays information about the current version and exits.

-help Displays information about the possible options and exits.

-env Displays the actual value taken for the options and exits. Allows you to verify that the option mechanism (explicit and implicit) is correctly set.

Default Options The previous o2options and others can be implicitly set as explained in 2.1.2.

Environment variables

O2HOME Is mandatory and contains the path to the O_2 installation directory.

O2OPTIONS See 2.1.2.

3

O_2 DBA Commands

O_2 DATABASE ADMINISTRATION COMMANDS

This chapter contains descriptions of O_2 database administration commands. To use these commands enter the **o2dba_shell** command. The **o2dba_shell** command is outlined in O2 System Commands.

These commands are also available when you use the **o2dsa_shell** command and all commands outlined in this manual are available if you have a development license.

Elements which can be administered are in subsections and listed in alphabetical order. Within each subsection commands are given in alphabetical order.

You will find at the end of this chapter the List of the various commands.

This chapter is divided into the following sections :

3 O2 DBA Commands

3.1 Applications

Commands exist to display O_2 application definitions. You can also display and print documentation in an application.

- ***Displaying an application***

To display program names and signatures of the current or specified application:

```
display application [appli_name]
```

To list all the applications defined in the current schema:

```
display applications
```

- ***Documentation for an application***

To display application documentation:

```
display doc in application [appli_name]
```

To print this documentation:

```
print doc in application [appli_name] path_name
```

- ***Printing an application***

To print the program names and signatures of the current or specified application:

```
print application [appli_name] path_name
```

- ***Running an application***

You can run an application using the following command:

```
run application appli_name
[(value_spec[, value_spec]...)]
```

This command firstly runs the application's program called **`init`** if it exists. It then displays a menu containing all public programs of the application. Otherwise the **`dashboard`** program is started if it exists. If the **`init`** program has parameters, you must specify the parameter values in a comma-separated list, enclosed between brackets **`()`** after the application name *`appli_name`*, in the same order as the **`init`** program parameters.

3.2 Bases

To manipulate bases, the following commands exist.

- ***Creating a base***

To create a base, use the command:

```
[create] base base_name [schema schema_name]
[volume volume_name] [size size] [factor factor]
```

This creates a logical database on the specified physical volume or the current volume, if no volume is specified.

The structure and behavior of the new base is determined by the schema *schema_name* (or by the current schema if no schema is specified).

An implicit **set base** is performed to set the current base to this new base. The name of the new base must not conflict with the name of any existing base.

A base and its schema can be on different physical volumes.

- ***Size***

An initial size in kilobytes can be specified for this database so that adjacent disk space is reserved for it. The system allocates extra space if this initial size is too small.

This size must be smaller than the remaining file space size of the volume. The initial default size is 64 kilobytes.

- ***Factor***

You can also specify a page fill factor between 1% and 100%. The default page fill factor is 95. This means that 5% of the page space is not used in order to accommodate expansions of objects (strings and collections).

This factor is only used when a persistent object is created and when the transaction works in the append mode. This mode can be activated by the O_2Engine API primitive, **o2_storage_layout** (O2_TAIL). See the *O_2Engine API Reference Manual* for further details.

- ***Deleting a base***

To delete a base, type the following command:

```
delete base [base_name]
```

This removes a base from the physical volume. All objects and values stored in the base are deleted. You cannot delete the current base as well as a base that has been set as the working base in the current transaction.

- ***Displaying bases***

To display the details of a base, or of the current base if no base is specified, use the following command:

```
display base [base_name] [stat]
```

Information displayed about the base includes its name, its internal identifier, the physical volume name on which it resides, the name of the schema to which it belongs.

When you use the *[***stat***]* option, you also obtain the number of O_2 files which make up the base and the size of the base. You also get information about each file contained in the base: the internal name and identifier of the O_2Store files, the cardinality of records in this file, the number of pages occupied on disk.

There are different kinds of files: **bagfile** for large sets, **collectionIndex** for large lists and unique sets, **o2Index** for user defined indexes, **cluster** for user defined clusters, **ldiFile** for long bytes and strings, and **defaultFile** for any other type of data.

To list the names of all bases, type the following command:

```
display bases [stat]
```

When you use the *[***stat***]* option, you obtain details for each base as outlined in the **display base** command above.

- ***Extend a base***

To add a volume to a base, use the following command:

```
extend base base_name volume volume_name
```

- ***Garbage a base***

To clean up a base use the following command:

```
garbage base [base_name]
```

This cleans up the base by collecting the unreachable objects of a given base and releases disk space. It is recommended that you use this command from time to time.

Ensure that all persistent objects in a base are reachable from a persistent root in a base. This is because garbage collection works within a base which means that those objects in a particular base that are only reachable from another base will also be garbaged.

- ***Renaming a base***

To rename a base, use the command:

```
rename base base_name as new_base_name
```

This changes the name of a specified base to the new name given, or of the current base if only the new base name is given.

The new name must not conflict with an existing base name.

- ***Resetting the index & cluster in a base***

To destroy invalid index and clusters after a schema has been modified, use the following command:

```
reset index cluster in base base_name
```

See the *O_2 System Administration Guide* for more details.

- ***Setting a base***

To set the base *base_name* to the current base, type:

```
set base base_name
```

The schema associated to the base is automatically set as the current working schema.

The physical volume containing the base is set as the current volume.

3.3 Clusters

Cluster commands exist for classes or named collections.

Create cluster

The **cluster** command syntax is as follows:

```
[ create ] cluster  root_name   [ on  path ]
                   [ index  index_path ]
                   [ in volume  vol_name ]
                   [ size  integer_number ]
                   [ at integer_number]
```

where *root_name* is

```
root_name  :=  constant_name  |  class_name
```

where *path* is

```
path := *[ (component_path) ] | (component_path)
component_path := attribute_name [*] [(component_path)]
         | component_path [, component_path...]
```

where *index_path* is

```
index_path  :=  attribute_name [. attribute_name  ...]
```

Options are outlined below.

- ***on***

  If you do not specify the path using the **on** clause, the class extent (i.e. all the persistent values of the class) or the complex value of a name, is added to the defined cluster.

- ***index***

  Cluster via an existing index definition given by *index_path*. The resulting cluster file is then sorted using the index.

- ***in volume***

  Cluster in the volume called *vol_name*. The default volume is the volume in which the base was initially created.

- ***size***

  Specify an initial size in kilobytes for the database in order to reserve adjacent disk space. The default size of an O_2 file is 32 kilobytes.

- ***at***

  Insert the cluster definition at position *integer_number* in the priority list. A low number corresponds to a high priority. If not specified, the cluster is inserted at the end of the list and will have the lowest priority.

- ***() and (,)***

  When applied to tuple structured objects or values, these brackets specify that components can be obtained using the attribute *attribute_name* and are to be clustered with their parents. Using a comma to separate them you can give several attributes, and for each one you can specify the components by *component_path*.

- *******

  When applied to collection-structured objects or values (set, bag or list), all the elements of the collection are clustered with that collection. Note that the elements themselves can be either collections or tuples, and you can enter another *component_path* for them.

Delete cluster

To delete the cluster given by *root_name*, use the following command:

```
delete cluster  root_name
```

Display clusters

To display the root names of all the clusters defined on the current base, use the command:

```
display clusters [stat]
```

If you use the **stat** keyword, the path and statistical information is also displayed.

Promote cluster

To promote newly created objects in clusters use the following command:

```
promote
```

This command triggers the clustering function but does not validate the transaction.

3.4 Display command

To display the names of the current schema, base and class, type the following command:

```
display [stat]
```

With **stat**, you also get statistical information about the current named system, the volume contained in the system, the size on disk, each schema name and owner, each base name and its associated schema, and the number of persistent collections and indexes for each base.

3.5 Help

To get on-line help, you use the command:

```
help [O2_keyword]
```

If you type **help**, you obtain a full list of O_2 commands. The syntax and a brief description is given for each command.

When you use an *O2_keyword*, **help** gives information for those commands which contain the keyword.

3.6 Indexes

You can maintain one or more indexes on a named set or list. This increases the speed at which set or list elements are accessed during query execution. You must declare the set or list as a *constant_name*.

- ***Creating an index***

To create and maintain an index for the named set or list *constant_name* on the path *index_path*, type the command:

```
[create] index constant_name on index_path
[in volume volume_name]
```

The *index_path* specification is as follows:

```
attribute_name [.attribute_name]...
```

The last *attribute_name* you give must be either an atomic value or an object. Any other *attribute_name*s are tuple-valued attributes.

- ***Displaying an index***

To list the names of all the collections on which indexes are defined

```
display index [stat]
```

To display the list of all the key paths for the collection *constant_name*:

```
display index constant_name [stat]
```

To display the details about the index specified:

```
display index constant_name on index_path [stat]
```

If you add **stat**, you get the index identifier, the cardinality and type of keys in the index, the type of indexed data, the number of pages and the internal O_2 Store file name.

- ***Deleting an index***

To delete an index for the named set or list on the path *index_path*:

```
delete index constant_name on index_path
```

3.7 Names

These commands apply to named objects and values.

- ***Displaying a named object or value***

To display the definition of the named object or named value specified:

```
display name obj_or_value_name [stat]
```

If you use **stat**, you get statistical information if the named object or value is a collection

To list all the names defined for the current schema:

```
display names
```

- ***Displaying documentation for a named object or value***

To display documentation on a named object or value:

```
display doc in name obj_or_value_name
```

- ***Printing a named object or value***

To print the definition of the named object or named value specified:

```
print name obj_or_value_name path_name [stat]
```

When you use the **stat** keyword, you get statistical information if the named object or value is a collection.

To print out the list of all the names defined for the current schema:

```
print names path_name
```

- ***Printing documentation for a named object or value***

To print documentation on a named object or value:

```
print doc in name obj_or_value_name path_name
```

3.8 Programs

- ***Displaying a program***

To display the signature of the program *prog_name* in the current or specified application:

```
display program prog_name [in application appli_name]
```

To list the names and signatures of all programs in the current or specified application:

```
display programs [in application appli_name]
```

- ***Documentation of a program (development option)***

To display documentation to the current or specified program:

```
display program doc [program_name]
[ in application appli_name]
```

- ***Printing a program (development option)***

To print the signature of the program *prog_name* in the current or specified application:

```
print program prog_name
[in application appli_name] path_name
```

To print this documentation:

```
print program doc [program_name]
[in application appli_name] path_name
```

To print the list of the names and signatures of all programs in the current or specified application:

```
print programs [in application appli_name] path_name
```

- ***Running a program***

You can run a particular program of an application using the command:

```
run program prog_name[(value_spec [, value_spec]...)]
in application appli_name [(value_spec [, value_spec]...)]
```

The program *prog_name* must be public. If the program has input arguments, you must specify the values in a comma-separated list, enclosed in brackets **()**, after the program name *prog_name*, in the same order as program arguments. If the application *appli_name* has an **init** program with parameters, you must specify the parameter values in a comma-separated list, enclosed in brackets **()** after the application name *appli_name*, in the same order as the **init** program parameters.

3.9 Query

To call the OQL query interpreter, type the command:

```
query
```

The interactive OQL interpreter is invoked, and you can enter query instructions. Refer to the *OQL User Manual* for more information.

To end a query session, type:

```
quit
```

This returns you to the main command interpreter.

3.10 Transactions

When you use O_2DBA you do so in the context of a transaction. To save your work you must validate the transaction. You can also roll back what you have done up to the last validation.

The following commands allow you to control the current transaction:

- ***Aborting a transaction***

```
abort
```

This cancels all the updates carried out during the current transaction and immediately starts a new transaction.

- ***Controlling commitment***

```
catalog transaction {on or off}
```

The default value is off which means that you control when you commit a transaction. When the value is set to on, O_2 automatically updates the catalogue in a single atomic transaction.

The following catalogue updates are carried out by O_2:

- **create/delete volume, base, index, cluster, schema**

- **import/export schema, base**

- **delete import/export**

- ***Committing a transaction***

```
commit
```

This ends the current transaction and commits modifications. It frees memory used by O_2 and starts a new transaction.

- ***Ending a transaction***

```
end transaction
```

This validates the current transaction and starts a read-only transaction. The next command you use will run without access to the locking mechanism. So updates cannot be carried out. This command is generally used before starting an OQL session without locking objects and without interfering with other transactions running concurrently.

- ***Quitting***

```
quit
```

This commits modifications, frees memory used by O_2 and ends the O_2 session.

- ***Returning to transaction mode***

```
transaction
```

This returns you to transaction mode when you have been in read-only mode.

- ***Validating a transaction***

```
validate
```

This ends the current transaction and commits modifications made but it does not free memory used by O_2. It then starts a new transaction.

3.11 Volumes

- ***Creating a volume***

To create a physical O_2 user volume, use the command:

```
[create] volume volume_name size integer
[blocksize integer] [in path_name]
```

This creates the volume *volume_name* with the specified size in kilobytes. The current volume is not set to this new volume. If the size is set to zero 80% of the available size of the file system is allocated for this volume. **blocksize** gives, in kilobytes, the size of the blocks to incrementally allocate storage on the disk. The default is 256 kilobytes. *path_name* is a directory where the volume will be created.

New schemas and bases can then be created in the new volume.

- ***Deleting a volume***

To delete a physical O_2 user volume, use the command:

```
delete volume [volume_name]
```

This removes an existing volume *volume_name*. Any schemas or bases stored on that volume must be deleted before deleting the volume otherwise an error is displayed. The volume must be deleted before mounting it. There is no explicit mount volume or dismount volume command, as volumes are mounted automatically. When a volume is deleted, the file in which it is stored is deleted. The volume **DefaultVol** cannot be deleted.

- ***Displaying volumes***

To list the volume name and the internal volume identifier, use the following command:

```
display volume [volume_name] [stat]
```

When you add **stat**, you obtain the volume size on disk. You also get the files corresponding to this volume in the catalogue directory. The volume size displayed is the size really occupied by the data and can be smaller that the corresponding operating system file.

To list all the physical O_2 user volumes associated with the current named O_2 system, type the following command:

```
display volumes [stat]
```

When you use **`stat`**, you obtain the information outlined in the **`display volume`** command for each volume.

- ***Extending a volume***

To extend an O_2 user volume with a new file, use the command

```
extend volume volume_name size integer [in pathname]
```

This adds to the volume an extension in the directory that is defined by the *`path_name`*. The size of the extension is given in kilobytes. If size = 0 80% of the available space in the file system is potentially usable.

- ***Renaming a volume***

To rename an existing physical O_2 user volume, use the command

```
rename volume volume_name as new_volume_name
```

This renames the volume *`volume_name`*. Note that only the name as known to O_2 is changed. The operating system name of the physical volume remains the same. The volume **`DefaultVol`** cannot be renamed.

3.12 List

The list of all the commands is given in alphabetical order :

- `abort`
- `catalog transaction {on or off}`
- `commit`
- `[create] base base_name [schema schema_name]`
- `[ create ] cluster  root_name [ on path ]`
- `[create] index constant_name on index_path [in volume volume_name]`
- `[create] volume volume_name size integer [blocksize integer] [in path_name]`
- `delete base [base_name]`
- `delete cluster root_name`
- `delete index constant_name on index_path`
- `delete volume [volume_name]`
- `display base [base_name] [stat]`
- `display bases [stat]`
- `display clusters [stat]`
- `display doc in application [appli_name]`
- `display doc in name obj_or_value_name`
- `display index constant_name [stat]`
- `display name obj_or_value_name [stat]`
- `display names`
- `display program prog_name [in application appli_name]`
- `display program doc [program_name]`
- `display programs [in application appli_name]`
- `display [stat]`
- `display volume [volume_name] [stat]`

3 O2 DBA Commands

- display volumes [stat]
- end transaction
- extend base base_name volume volume_name
- extend volume volume_name size integer [in pathname]
- garbage base [base_name]
- help [O2_keyword]
- print application [appli_name] path_name
- print doc in application [appli_name] path_name
- print doc in name obj_or_value_name path_name
- print name obj_or_value_name path_name [stat]
- print names path_name
- print program prog_name
- print program doc [program_name]
- print programs [in application appli_name] path_name
- promote
- query
- quit a query
- quit a transaction
- rename base base_name as new_base_name
- reset index cluster in base base_name
- run application appli_name
- run program prog_name[(value_spec [, value_spec]...)]
- set base base_name
- transaction
- validate

4 O_2 DSA Commands

O_2 SCHEMA ADMINISTRATION COMMANDS

This chapter provides descriptions of O_2 schema administration commands. To use these commands enter the **o2dsa_shell** command. The **o2dsa_shell** command is outlined in O2 System Commands.

You can also use all O_2DBA commands outlined in O2 DBA Commands.

These commands are also available directly under O_2 shell if you have an O_2C development license.

Some commands are only available with a development license.

Elements which can be administered are in subsections and listed in alphabetical order. Within each subsection commands are given in alphabetical order.

You will find at the end of this chapter the List of all the various commands.

This chapter is divided into the following sections :

- Attributes
- Bases
- Classes
- Libraries
- Methods
- Names
- Schemas
- Sources

4 O2 DSA Commands

4.1 Attributes

Tuple attributes can be updated. Only the rename command is outlined here.

- ***Renaming an attribute***

To rename an attribute in the current or specified class:

```
rename attribute attribute_name
[from class superclass_name]
as attribute_name [in class class_name]
```

Only use the **`from class`** clause for inherited attributes.

4.2 Bases

You can also use all the O_2DBA commands which apply to bases. See Section 3.

- ***Importing a base***

To access information stored in a base of another schema, you use the command:

```
 import base base_name
        name object_name [, object_name]...
```

Your application can now access data stored in your base and data stored in another base through the imported names. This command specifies various named object definitions defined in the schema governing the specified base. These definitions must be marked as exportable.

Each method applied to an object belonging to another base runs on this base. If such a method creates new persistent objects, these objects are created in the imported base.

- ***Modifying base status***

To modify the status of a base, use the following command:

```
[modify] status TEST [in base base_name]
```

4.3 Classes

Commands exist to delete and display class definitions and documentation, to rename a class and to rename an attribute of a class.

You can also use all the O_2DBA commands which apply to classes. See Section 3.

- ***Confirming class updates (development option)***

When you change the physical structure of a class, when you create a new class or when you import a class from another schema you must confirm the updates or creation before opening or creating related bases.

```
confirm class [class_name]
```

```
confirm classes
```

- ***Deleting a class (development option)***

To delete the definition of the current class or of the `class_name` specified, use the command:

```
delete class [class_name]
```

To delete the definition of the specified class *class_name* and all its subclasses, you can use the command:

```
delete classes from class_name
```

- ***Displaying a class***

To display the type definition of the current class or specified class *class_name*, use the command:

```
display class [class_name]
```

To list the names of all classes defined under the current schema, you can use the command:

```
display classes
```

- ***Display documentation about a class***

To display documentation about the current or specified class, use the command:

```
display doc in class [class_name]
```

- ***Documentation about a class***

To add documentation about the current or specified class, use the command:

```
doc in class [class_name] [documentation]
```

- ***Inheritance of a class***

You can change the superclass of a class:

```
[create] inherit [Class_name] Superclass_name
```

This command makes the specified or current class inherit its properties from the superclass *Superclass_name*. This implies possible changes to the type and method definitions of the class.

You can delete a link between a superclass and a class:

```
delete inherit [Class_name] Superclass_name
```

This means that the specified or current class no longer inherits its properties from the superclass *Superclass_name*. This implies possible changes to the type and method definitions of the specified class. If the specified class has no other superclasses, it becomes a direct subclass of the class **Object**.

- ***Printing a class***

To print the type definition of the current class or specified class *class_name*, use the command:

```
print class [class_name] path_name
```

To print the list of names of all classes defined under the current schema, use the command:

```
print classes path_name
```

- ***Printing documentation on a class***

To copy the documentation to a file, use the command:

```
print doc in class [class_name] path_name
```

- ***Reimporting a class***

To reimport a class after it has been modified, use the command:

```
reimport class class_name
```

- ***Renaming a class***

To rename the current or specified class, use the command:

```
rename class [class_name as] new_class_name
```

- ***Saving a class***

To save a class source definition in a file:

```
save class [class_name] path_name
```

4.4 Libraries

Binary objects are produced when you compile O_2C code or when you import C++ function members. You can create libraries to contain all the binary objects of a schema. The contents of these libraries are stored in an O_2 database and are transformed into libraries when the schema is opened. You can also store a library in a standard file.

The following library commands cannot be used on Sparc stations running SunOS.

When a library is stored in an O_2 schema, it is loaded the first time one of its methods is used. If the library is stored in a file it becomes external to O_2. You must call it in a Makefile or when you start O_2 with the **-libpath** and **-libs** options.

The directory where a library is to be stored must exist. The library is not created if the path name for the file exists already.

- ***Creating a library***

To create a library containing all the binary objects belonging to a schema called *[schema_name]*, use the following command:

```
create library [schema_name] [path_name]
```

The library is stored in the actual O_2 schema used to build it, or in the specified file (if specified).

This command works even if all the schema definitions are not complete. It does not work if the source code of a program/ method/ function is missing.

- ***Creating a shared library***

To create a shared library, use the following command:

```
create shared library [schema_name] [path_name]
```

This is the same as the create library command except that the library is shared between different schemas giving increased performance and disk space.

Note that the binaries used to create a shared library are deleted from the schema. So that if you need to change O_2C code, or import new C++ functions, or change the signature of an imported C++ function, you must rebuild all binaries. This implies a complete recompilation of the schema. This command is recommended for schemas that are stable and are not often modified.

4.5 Methods

Methods include O_2C methods and imported C++ function members.

- ***Displaying a method***

To display the signature of a method *method_name*:

```
display method method_name [in class class_name]
```

To display the names and signatures of the locally-defined methods in the current or specified class:

```
display methods [in class class_name]
[from superclass_name]
```

Use the **from** option to display all the methods inherited from the superclass *superclass_name*.

- ***Documentation of a method***

To add documentation to the specified method:

```
method doc [method_name] [in class class_name]
[documentation]
```

To display this documentation:

```
display method doc [method_name] [in class class_name]
```

To print this documentation:

```
print method doc [method_name] [in class class_name]
path_name
```

- ***Printing a method***

To print the signature of a method *method_name*:

```
 print method method_name [in class class_name]
```

To print the names and signatures of the locally-defined methods in the current or specified class:

```
 print methods [in class class_name]
[from superclass_name] path_name
```

Use the **from** option to print all the methods inherited from the superclass *superclass_name*.

4.6 Names

Commands exist to create, delete and display name definitions and documentation as well as to modify and change names.

You can also use all the O_2DBA commands which apply to names. See Section 3.

- ***Creating a named object or value***

To create a named object or named value, use the command:

```
[create] [constant] name obj_or_value_name: type_spec
```

If *type_spec* is a class, it must exist in the schema.

If the name is constant or if *type_spec* is a type, an instance of an object or value, *type_spec* is created in the database.

For a non-constant object name, only the name is created.

The object itself must be created by a program using the `new` instruction.

- ***Deleting a named object or value***

To delete a name:

```
delete name obj_or_value_name
```

This does not automatically delete the actual value or object.

If the value or object is persistent for another reason, for example. if it is part of another persistent object, the actual object or value is not deleted.

- ***Documentation for a named object or value***

To add documentation to a current or specified named object or value:

```
doc in name [obj_or_value_name] [documentation]
```

- ***Modifying a named object or value***

To modify a named object or named value:

```
modify [constant] name obj_or_value_name: type_spec
```

You can make a name constant or remove the constant property, in which case the *type_spec* is the same as in the original definition and the data referred to by this name can still be accessed.

However, if the type is changed, the link between the name and data is broken and the command works in exactly the same way as the create name command.

- ***Renaming a named object or value (development option)***

To change the name associated with an object or value:

```
rename name obj_or_value_name as obj_or_value_name
```

This does not affect the actual object or value. If the name is a constant name, the new name is also constant.

- ***Saving a name (development option)***

To save a name source definition to a file:

```
save name obj_or_value_name path_name
```

4.7 Schemas

- ***Creating a schema***

To create a schema, use the command:

```
[create] schema schema_name [volume volume_name]
```

This creates a schema *schema_name* on the physical volume *volume_name* (or the current volume, if no volume is specified). The new schema only contains the definition of the system-supplied class **Object**.

An implicit **set schema** occurs to set the current schema to the new schema. The *schema_name* must not already exist for an existing schema or base.

- ***Deleting a schema***

To delete a schema, use the command:

```
delete schema [schema_name]
```

This removes the specified schema from the physical volume. This operation is refused if there are bases associated with the schema. The current schema cannot be deleted.

A schema cannot be deleted if it is set as the working schema in the current transaction. The best way is to delete the schema as the first command of the session.

Alternatively, you can move to another schema using the **set schema** command, commit the transaction and then delete the schema. If the schema you move to has been created in a different volume to the schema you want to delete, you remain positioned in that volume after the commit, and this may cause errors. For example, moving to the predefined schema **o2kit** is not recommended as it is created in the catalogue volume.

- ***Displaying schemas***

To display the details of the specified schema, or of the current schema if no schema is specified, use the following command:

```
display schema [schema_name] [stat]
```

Information displayed about the schema includes its name, its internal identifier, the physical volume name on which it is found, the number of bases that it contains.

When you use the **stat** option, you obtain more information. You also get a list of all user bases in this schema, the number of classes and methods associated with this schema, the size of the source base and definition base. Each schema is made up of two bases: the first contains the schema sources and the second contains the compiled schema.

To list the names of all schemas under the present physical volume, type the following command :

```
display schemas [stat]
```

When you use the **stat** option, you obtain more information for each schema as outlined in the **display schema** command above.

- ***Displaying schemas with imported elements***

To list schemas that have imported elements from the schema *schema_name*, you use the command:

```
display import schema_name
```

To list all the schemas that have imported elements from the base *base_name*, use the command:

```
display import base_name
```

- ***Deleting an export command***

To reverse an export command:

```
delete export [class class_name [, class_name]...]
[name obj_or_value_name [, obj_or_value_name]...]
```

If no schema definitions are specified, the command is refused.

- ***Exporting a schema***

You can use all or part of another schema to build your schema by importing one or more of its classes.

To export one or more schema definitions, you use the command:

```
export schema [class class_name [, class_name]...]
[name obj_or_value_name [, obj_or_value_name]...]
```

Each specified class and name definition of the current schema is marked as being exportable. This means that these definitions are now public and you can import them from another schema.

The simple command **export schema** refreshes the exported items after an update of the exported classes.

- ***Importing a schema***

To import all or some of another schema's definitions, use the command:

```
import schema schema_name
     [class class_name [, class_name]...]
     [name obj_or_value_name [, obj_or_value_name]...]
```

In the current schema, your application can now use the class, named object and named value definitions specified as if they had been defined in the current schema.

However, you cannot modify these imported class definitions. You must create subclasses and modify the methods and type structures of these subclasses. An imported class definition always has a private type structure even if it has a public structure in the schema of origin. If necessary, you can use the **public** or **read** options of the proper commands to change this.

- ***Renaming a schema***

To rename a schema, use the following command:

```
rename schema schema_name as new_schema_name
```

This changes the name of the specified schema, or of the current schema if only one schema name is given.

The new name must not conflict with any existing schema.

- ***Saving a schema (development option)***

To save the definitions of the schema *schema_name* to one or more ASCII files, type the command:

```
save schema [schema_name] path_or_file_name
```

If the *path_or_file_name* corresponds to a directory, the schema definitions are stored, one by one, in separate files:

- Each class definition is stored in two files:

  schema_name.Class_name.**o2**, for the class structure and method signatures, and *schema_name.Class_name*.**o2c**, which holds the method bodies.
- Each application is stored in two files:

  schema_name.appli_name.**o2**, containing the application definition, the variable definitions and the program signatures, and *schema_name.appli_name*.**o2c**, which holds the program bodies.
- All function definitions and signatures are stored together in a file called *schema_name.***funcs.o2**, and their bodies are stored in *schema_name.***funcs.o2c**.
- All named type definitions are stored together in a file called *schema_name.***types.o2**.
- All named object and value definitions are stored in a file called *schema_name.***names.o2**.
- All redefined (method or attribute) properties are stored in the file *schema_name.***properties.o2**
- Three command files are generated to restore the schema definitions:
  - *schema_name.***load.o2** restores all schema definitions (i.e. files ending in **.o2**, but none of the O_2C implementations.
  - *schema_name.***load.o2c** restores the bodies (files ending in **.o2c**).
  - *schema_name.***load** restores everything.

These command files may be invoked under an O_2 shell in the following way:

#"*[directory_path]schema_name.load[.o2[c]]*"

If *path_or_file_name* corresponds to a file, or anything other than a directory, then the entire schema is stored in a single file with that name. This file may then be invoked to restore the schema as a whole (or to reproduce it in another schema elsewhere) by executing it as an O_2 command file:

#"*directory_path_or_file_name*"

- ***Setting a schema***

To set the schema *schema_name* to the current schema, use the command:

```
set schema schema_name
```

The physical volume that contains the schema is set as the current volume.

4.8 Sources

- ***Creating sources***

If you have created your schema in O_2Engine API and you want to visualize it in O_2Tools you can use this command to create the sources:

```
create sources
```

You also use this command if you want to reestablish your sources after having destroyed them using the **`delete sources`** command.

Important

This command only re-establishes the source of the classes not the bodies of methods, programs, functions, documentation or cross references.

- ***Deleting sources (development option)***

You use this command to destroy all the sources of the current schema.

```
delete sources
```

4.9 Inheritance

- ***Creating inheritance***

You can set up inheritance links between two classes that already exist using the command :

```
 inherit subclass_name superclass_name
```

- ***Deleting inheritance***

You can also destroy inheritance links using :

```
delete inherit subclass_name superclass_name
```

The subclass stops inheriting the properties of the superclass.

4.10 List

- `confirm class [class_name]`
- `confirm classes`
- `[create] inherit [Class_name] Superclass_name`
- `create library [schema_name] [path_name]`
- `[create] [constant] name obj_or_value_name: type_spec`
- `[create] schema schema_name [volume volume_name]`
- `create shared library [schema_name] [path_name]`
- `create sources`
- `delete class [class_name]`
- `delete classes from class_name`
- `delete export [class class_name [, class_name]...] [name obj_or_value_name [, obj_or_value_name]...]`
- `delete inherit [Class_name] Superclass_name`
- `delete name obj_or_value_name`
- `delete schema [schema_name]`
- `delete sources`
- `delete inherit subclass_name superclass_name`
- `display class [class_name]`
- `display classes`
- `display doc in class [class_name]`
- `display import base_name`
- `display import schema_name`
- `display method method_name [in class class_name]`
- `display method doc [method_name] [in class class_name]`

- `display methods [in class class_name] [from superclass_name]`
- `display schema [schema_name] [stat]`
- `display schemas [stat]`
- `doc in class [class_name] [documentation]`
- `doc in name [obj_or_value_name] [documentation]`
- `export schema [class class_name [, class_name]...] [name obj_or_value_name [, obj_or_value_name]...]`
- `import base base_name`
- `import schema schema_name  [class class_name [, class_name]...]  [name obj_or_value_name [, obj_or_value_name]...]`
- `inherit subclass_name superclass_name`
- `method doc [method_name] [in class class_name] [documentation]`
- `modify [constant] name obj_or_value_name: type_spec`
- `[modify] status TEST [in base base_name]`
- `print class [class_name] path_name`
- `print classes path_name`
- `print doc in class [class_name] path_name`
- `print method method_name [in class class_name]`
- `print method doc [method_name] [in class class_name] path_name`
- `print methods [in class class_name] [from superclass_name] path_name`
- `reimport class class_name`
- `rename attribute attribute_name`
- `rename class [class_name as] new_class_name`
- `rename name obj_or_value_name as obj_or_value_name`

- `rename schema schema_name as new_schema_name`
- `save class [class_name] path_name`
- `save name obj_or_value_name path_name`
- `save schema [schema_name] path_or_file_name`
- `set schema schema_name`

4 O2 DSA Commands

INDEX