

Creative assets management



Extensis™

Portfolio<sup>TM</sup>8

AppleScript  
Guide

# Contact

## Extensis

1800 SW First Avenue,  
Suite 500  
Portland, OR 97201  
Toll Free: (800) 796-9798  
Phone: (503) 274-2020  
Fax: (503) 274-0530  
<http://www.extensis.com>

## Extensis Europe

First Floor, Century House  
The Lakes  
Northampton NN4 7SJ  
United Kingdom  
Phone: +44(0)1604 636 300  
Fax +44 (0)1604 636 366  
[info@extensis.co.uk](mailto:info@extensis.co.uk)

© 2006 Extensis, a division of Celartem, Inc. This document and the software described in it are copyrighted with all rights reserved. This document or the software described may not be copied, in whole or part, without the written consent of Extensis, except in the normal use of the software, or to make a backup copy of the software. This exception does not allow copies to be made for others. Licensed under U.S. patents issued and pending.

Extensis is a registered trademark of Extensis. The Extensis logo, Extensis Library, Font Reserve, Font Reserve Server, Font Vault, and Font Sense, Portfolio, Portfolio Server, Portfolio NetPublish, NetPublish, Suitcase and Suitcase Server are all trademarks of Extensis. Celartem, Celartem, Inc., the Celartem logo, PixelLive and PixelSafe are trademarks of Celartem, Inc. Adobe, Acrobat, Illustrator, Photoshop, and PostScript are trademarks of Adobe Systems, Incorporated. Apple, AppleScript, Bonjour, FontSync, Macintosh, Mac OS 9, Mac OS X, PowerPC, and QuickDraw are registered trademarks of Apple Computer, Inc. Microsoft, Internet Explorer, Windows, Windows XP, Windows 2000, Windows NT, Windows ME and Windows 98 are registered trademarks of Microsoft Corporation. Intel is a registered trademark of Intel. All other trademarks are the property of their respective owners.

Portions of this product use software components developed through various open source projects. The licenses and availability of source

## Celartem, Inc.

Email: [sales\\_ap@celartem.com](mailto:sales_ap@celartem.com)  
<http://www.celartem.com/jp/>

## Press Contact

Phone: (503) 274-2020 x129  
Email: [press@extensis.com](mailto:press@extensis.com)

## Customer Service

Web/email: <http://www.extensis.com/customerservice/>  
Phone: (800) 796-9798

## Technical Support

Web/email: <http://www.extensis.com/support/>

## Documentation Feedback

Web/email: <http://www.extensis.com/helpfeedback/>

code for such components are specified in the copyright notice file, LICENSES.TXT delivered with this product. Please refer to these licenses for information regarding use of these software components.

Extensis warrants the disks on which the software is recorded to be free from defects in materials and faulty workmanship under normal use for a period of thirty (30) days from the original date of purchase. If you purchased this product directly from Extensis, and if a defect occurs during the 30-day period, you may return the disks to Extensis for a free replacement. All products submitted for replacement must be registered with Extensis before replacement. Extensis products purchased from resellers are warranted by the reseller and are covered by the reseller's return policy. This warranty is limited to replacement and shall not encompass any other damages, including but not limited to loss of profit, and special, incidental, or other similar claims. This software is provided on an "as is" basis. Except for the express warranty set forth above, Extensis makes no other warranties, either explicit or implied, regarding the enclosed software's quality, performance, merchantability, or fitness for a particular purpose.



# Contents

<b>AppleScript and Portfolio</b> .....	<b>4</b>
<b>Upgrading scripts written for prior versions of Portfolio</b> .....	<b>5</b>
The Document Class.....	5
Cataloging Options .....	5
Make Commands .....	6
Getting Property Values .....	6
The Collect Command .....	7
The Make Custom Field Command.....	7
The File Rename Settings Class.....	8
<b>How to Automate Portfolio with AppleScript</b> .....	<b>9</b>
Starting Portfolio .....	9
Working with Catalogs .....	9
Cataloging Options .....	10
Working with Galleries .....	11
Adding files to a Gallery .....	12
Working with Records .....	13
Working with Fields .....	14
Searching the Catalog .....	16
<b>The Portfolio AppleScript Dictionary</b> .....	<b>18</b>

# AppleScript and Portfolio

This document contains a brief overview of the Portfolio commands available via the scripting interface in Extensis Portfolio for the Macintosh. The examples shown here were done in AppleScript, but users familiar with other OSA-compliant languages should be able to extrapolate from the examples for their particular languages.

A basic level familiarity with AppleScript is assumed. For a more thorough introduction to AppleScript, we recommend visiting the Apple website, <http://www.apple.com/applescript/>. This site contains a great deal of useful information about AppleScript as well as links to other scripting-related web sites.

For more information about Portfolio, please refer to the Portfolio User Guide that is installed with your Portfolio client. For the most up-to-date information about Portfolio, please visit the Extensis website at <http://www.extensis.com/portfolio/>.

# Upgrading scripts written for prior versions of Portfolio

This release of Portfolio include many feature upgrades as well as preexisting features that might require scripts from Portfolio 6 or earlier to be updated. The following items were changed in the Portfolio 7 release, and are still relevant in the current release.

## The Document Class

Portfolio 7.0 changed the relationship between catalogs and galleries. Catalogs are now represented by the `document` class. This is the new top-level container class in Portfolio. A document has one or more galleries.

Preexisting scripts must be modified so that the top-level container objects of commands and clauses are instances of a document rather than a gallery. This can most easily be accomplished by wrapping sections of old scripts in a `tell document` clause.

### Example:

Portfolio 6:

```
get name of record 1 of gallery 1
```

Portfolio 7 and beyond:

```
tell document 1
  get name of record 1 of gallery 1
end tell
```

## Cataloging Options

Starting in Portfolio 7, catalogs have one or more sets of cataloging options presets, obtainable by index. You can get the user's default cataloging options by asking the catalog for its `user_options` property. You can also get the currently used set of cataloging options by asking the catalog for its `current_options` property.

### Examples:

```
get properties of cataloging options 2
of document 1
```

```
get properties of user_options of
document 1
```

```
get properties of current_options of
document 1
```

In addition, many properties of cataloging options have been separated out into a new class called `advanced cataloging options`; see [pages 18-26](#) for complete information.

## Make Commands

Starting in Portfolio 7, in order to maintain compliance with Apple's guidelines, new commands are now make commands.

### Example:

Portfolio 6:

```
new catalog file "Macintosh HD:Desktop
Folder:Untitled.fdb"
```

```
new record using name "q"
```

Portfolio 7:

```
make catalog "Macintosh HD:Users:me:
Documents:Untitled.fdb" with previews no
```

```
make record with name "q"
```

## Getting Property Values

Starting in Portfolio 7, to get the value of a property, you must use the keyword `value` explicitly.

### Example:

Portfolio 6:

```
get field "Filename" of record 1
```

Portfolio 7:

```
get value of field "Filename" of record 1
```

Similarly, to get the name of a record, do so explicitly:

### Example:

Portfolio 6:

```
get record 1
```

Portfolio 7:

```
get name of record 1
```

## The Collect Command

The `collect` command allows you to automate the collection of files in a gallery to a new location. The only required parameters are the gallery to collect, and the location of the destination directory. A destination directory must exist; the `collect` command does not create this directory. The optional parameters correspond to the options in Collect dialog box; see [pages 22-23](#) of this guide for a full list of parameters.

### Example:

```
-- create a destination directory "Mac
HD:Users:me:Desktop:collectedCatalog"
```

```
tell application "Portfolio"
    tell gallery 1 of document 1
        collect to location alias "Mac HD:
Users:me:Desktop:collectedCatalog:"
        create archive catalog "Mac HD:Users:
me:Desktop:collectedCatalog:catalog.
fdb"
        preserve file hierarchy yes
        using preview mode size 480
        create mac browser yes
    end tell
end tell
```

## The Make Custom Field Command

The `make custom field` command allows you to create new custom fields. There are three required parameters:

1. *The document to which you want to add the new field*
2. *The name of the new field*
3. *The type of the new field.*

The optional parameters correspond to options in Portfolio's New Custom Field dialog; see [page 24](#) of this guide for a full list of parameters.



*You must have an administrator access level to use this command.*

### Example:

```
tell application "Portfolio"
    tell document 1
        make custom field with name "a date" of
type DateTimeType
        make custom field with name "a decimal"
of type DecimalType
        make custom field with name "a number"
of type NumberType
        make custom field with name "a text
block" of type TextBlockType
        make custom field with name "a text" of
type TextType
        make custom field with name "a web url"
of type WebURLType
    end tell
end tell
```

## The File Rename Settings Class

You can specify the settings Portfolio applies when renaming files during cataloging by setting the file rename settings of the cataloging options class. The settings correspond to the options in the Cataloging Options dialog box; see [page 20](#) of this guide for a complete list of the settings available.

### Example:

```
tell application "Portfolio"
  tell document 1
    set rename before cataloging of
      current_options to true
    tell file rename settings of current_
      options
      set front mode to text
      set front text to "prefix_"
    end tell
  end tell
end tell
```

# How to Automate Portfolio with AppleScript

## Starting Portfolio

As with any application in AppleScript, use the `tell` command to address Portfolio. If Portfolio is not active, the application will launch. If it is active, Portfolio will become the frontmost application.

### Example:

```
tell application "Portfolio"
    activate
end tell
```

To determine whether Portfolio is already running, you can poll the Finder to determine if Portfolio exists in the list of active processes.

### Example:

```
tell application "Finder"

    set P_app to every process whose name
    is "Portfolio"

    if P_app is {} then
        -- Because the list is empty,
        Portfolio is not running.
    end if
end tell
```



*You will probably want to turn off the "Show Open Dialog at Startup" option in the General pane of the Preferences dialog. Leaving this option on will bring up a dialog each time the application is launched.*

## Working with Catalogs

### Opening catalogs

To open a Portfolio catalog via AppleScript, use the `open catalog` command. If the catalog has already been opened with this copy of Portfolio (either through scripting or manually), the catalog will come to the front and become the active catalog.

### Example:

```
open catalog alias "Macintosh HD:Users:
me:Documents:Test.fdb"
```

To specify a particular Access level with which to open the catalog, use the `using mode` attribute. To specify a password, use the `with password` attribute. To specify a user name, use the `with username` attribute.

### Example:

```
open catalog alias "Macintosh HD:
Users:me:Documents:Test.fdb" using mode
publisher with password "mypassword"
```

## Opening a catalog on a server

To open a Portfolio catalog, which is being served by a Portfolio Server, use the **open server** command. The command takes the **address** of the catalog in the form **server/catalog**, where “server” is the server’s name or IP address.

### Example:

```
open server "portfolio://10.1.5.81:1313/
Test.fdb"
```

To specify a particular Access level with which to open the catalog, use the **using mode** attribute. To specify a password, use the **with password** attribute. To specify a user name, use the **with username** attribute.

### Example:

```
open server "portfolio://10.1.5.81:131
3/Test.fdb" using mode publisher with
password "mypassword"
```

## Creating catalogs

To create a new catalog, use the **new catalog** command. By default, the catalog opens in Administrator mode. To have the catalog generate screen-resolution previews use the **with previews** attribute.

### Example:

```
make catalog "Macintosh HD:Users:me:
Documents:Test.fdb" with previews yes
```

## Cataloging Options

A catalog may have zero or more preset Cataloging Options, as well as the user’s default Cataloging Options. To get the current options, use the **current \_ options** property of a catalog.

### Example:

```
get properties of current _ options of
document 1
```

Several of the options in the Cataloging Options dialog have been exposed to the scripting interface. In particular, the copy or move, assign properties, and description attributes may be specified.

To set these properties, use the **cataloging options** object. These attributes can all be set at once by accessing the **properties** record, or each individual property can be set.

### Examples:

```
set properties of user _ options to
{thumbnail size:size256}

set assign properties of user _ options to
no
```

## Working with Galleries

### Selecting a Gallery

Many commands require a reference to a Gallery object. This tells Portfolio not only which catalog's records are to be manipulated, but also (in the case of a record index) which visible records are being manipulated. In most of the following examples (particularly for records), each command has a reference to the desired Gallery object. Galleries can be referenced by index or by name.

#### Examples:

```
of selected gallery -- Currently selected gallery
```

```
of gallery 5 -- Gallery with index of 5
```

```
of gallery "Beach Photos" -- Gallery titled 'Beach Photos'
```



*These are just clauses, not actual commands. See the following sections for examples of usage with other commands.*

### Sorting a Gallery

Use the `sort` command to put the records in a gallery in a particular order. Be sure to only use indexed, single-valued fields for sorting.

#### Example:

```
tell gallery 1 of document 1 to sort by "Filename"
```

```
tell gallery 1 of document 1 to sort by "Created" order ascending
```

### Making a new Gallery

To add a gallery to a catalog, use the `make gallery` command.

#### Example:

```
tell document 1 to make gallery with name "foo"
```

### Displaying a Gallery

To display a gallery, use the `display` command.

#### Example:

```
tell document 1 to display gallery "foo"
```

## Adding files to a Gallery

To add source files to a gallery, use the `catalog` command. The `catalog` command observes all the settings set in the Cataloging Options dialog in the Portfolio application.

### Example:

```
catalog alias "Macintosh HD:Library:
Desktop Pictures:"
```

Any file specification can be used with the `catalog` command; this makes it very simple to catalog folders or entire volumes, as well as individual files.

Cataloging processes in Portfolio are independent, threaded processes. As soon as the process begins, control will return to the AppleScript. This prevents the script from timing out and allows the script to continue other operations while the cataloging process continues (as this may be a time-consuming process if there are many files). However, there are times when the script will want to manipulate the records that were added to the catalog as a result of the cataloging process. In these situations, the script needs to wait for the cataloging process to complete. To accomplish this, poll the `is cataloging` flag in Portfolio to determine whether a cataloging operation is in progress. A “true” result indicates that Portfolio is still cataloging and indicates steps should be taken within the script to pause until the value changes to “false”.

### Example:

```
repeat while is cataloging
delay 1
end repeat
```

## Cataloging Options

Several of the options in the Cataloging Options dialog have been exposed to the scripting interface. To set these properties, use the cataloging options object. These attributes can all be set at once by accessing the `properties` record, or each individual property can be set.

### Examples:

```
set properties of user_options of front
document to {description:"qwerty", assign
properties:true}

set description of user_options of front
document to "qwerty"
```

## Working with Records

Each gallery typically contains one or more record objects (though it may also contain no records). Use the record objects to iterate through a particular set of records to manipulate the values of the fields within each record.

Records can be referenced by index (the default), by ID, by name, etc. Be aware that an index represents the current order of records within a particular gallery, so a particular record's index will change based on the contents and order of the gallery. By contrast, the record ID is unique throughout the catalog and is not affected by the display within the current gallery.

### Example:

```
get name of record 1
get name of record id 803
get name of records 1 through 6
```

## Making a new Record

To make a new placeholder record, use the `make record` command.

### Example:

```
tell document 1 to make record with name
    "new record"
```

## Copying Records

To copy a record from one gallery to another, use the `gallery copy` command.

### Examples:

```
-- copy a single record
gallery copy record 1 of gallery 1 of
document 1 to gallery 3 of document 1

-- copy a range of records
gallery copy records 4 through 8 of
gallery 1 of document 1 to gallery 3 of
document 1

-- copy a single record between
documents
gallery copy record 1 of gallery 1 of
document 1 to gallery 1 of document 2
```

## Displaying a Record

To display a record use the `display` command. The record will scroll into view.

### Example:

```
tell gallery 1 to display record 144
```

## Working with Selections

A common use of scripts is to perform operations based on the selection of records the user has made in the catalog. To determine which records are currently selected, simply refer to the selection object within a particular gallery object.

### Example:

```
get ID of selection of gallery 1
```

## Selecting items

To modify the selection, use the `select` command. The `select` command does not deselect the current selection, so it may be necessary to use the `unselect` command to clear the selection first (see below).

### Examples:

```
tell gallery 1 of document 1
    select (records 10 through 13)
    select record id 68
    select every record
end tell

select every record of gallery 1 of
document 1 whose name starts with
"Abstract"
```

## Clearing the selection

Use the `unselect` command to clear the entire selection.

### Example:

```
unselect all of gallery 1 of front
document
```

## Working with Fields

Each Record object contains a number of `field` objects (one for each system field and one for each custom field). In addition, the Record object also contains a property for each system field. This means that system fields can be accessed through two different mechanisms; either as a property of the record or by accessing the particular field object. Either interface can be used; the results will not differ. Custom fields, on the other hand, cannot be presented as properties of the record (as they are not standard), and therefore can only be accessed through the appropriate field object.

To access system fields through the Record properties, simply refer to the appropriate property of the record.

### Example:

```
get filename of record 1 of gallery 1 of
document 1
```

To access any field in the catalog through the catalog object, you must use the `field` object. Field objects can be accessed via either the name or the index of the field.

### Examples:

```
tell gallery 1 of document 1
    get filename of record 1
    get value of field "filename" of record
    1
    get value of field 25 of record 1
end tell
```

## Determining the fields for a catalog

Because Portfolio allows the user to customize the catalog by creating their own fields, it is often useful to determine which fields are present in a particular catalog. To do this, simply request a list of all the field names for a particular record (any record within a catalog will return the same results).

### Example:

```
set lFields to the name of every field of
record 1 of gallery 1 of document 1
```

This returns a list of all fields in the catalog. It is then a simple matter to determine if a particular field exists within the catalog.

### Example:

```
if lFields does not contain "FieldX" then
-- 'FieldX' does not exist in the catalog
end if
```

## Changing Field Values

One of the most powerful aspects of the Portfolio scripting interface is the ability to not only read all of the fields in a record, but also to modify all of the fields (including the system fields). Be aware that while this is a very useful tool, it is also possible to ruin a catalog by incorrectly modifying data that Portfolio uses to manage source files. For example, incorrectly modifying the path of each record in a catalog could result in Portfolio being unable to find any records again.

Setting the field value is done in the same manner as getting the field values. Again, with system fields, the value can be accessed either via the field object or via the property in the record.

### Examples:

```
set value of field "Filename" of record 1
of gallery 1 of document 1 to "This is a
test description"
```

```
set filename of record 1 of gallery
1 of document 1 to "This is a test
description"
```

## Notes on Setting Field Values

### Data Types

When getting or setting field values, typing depends on whether you are referring to a field as a value or by name. If referring to a field value by name, field values should be passed to Portfolio as strings, regardless of the data type of the field within Portfolio. Passing the value as anything else will result in an error. Custom fields must be referred to by name, since they are not built-in fields.

If referring to a built-in field value by record property, field values should be passed to Portfolio as the appropriate type.

### Examples:

```
tell record 1 of gallery 1 of document 1

set value of field "Modified" to
"Friday, October 1, 2004 12:00:00 AM"

set last modified to date "Friday,
October 1, 2004 12:00:00 AM"

end tell
```

In the latter case, if a variable already exists as another data type, the value must be coerced to a string when passing it to Portfolio.

**Example:**

```
set dUpdate to date "Monday, December 1,  
1997 12:00:00 AM"
```

```
set value of field "Modified" of record 1  
of gallery 1 of document 1 to (dUpdate  
as string)
```

**Multi-valued Fields**

For setting multi-valued fields, there are two different methods which can be used. The first method is used to add a single value to the multi-valued list. To do so, simply set the field to a single value. In the following example, the keyword "Testing" is appended to the pre-existing list of values. If the value already exists in the list, it is not added (as a list in Portfolio cannot contain duplicate values).

**Example:**

```
set keywords to "Testing"
```

The other method is used to replace the entire multi-valued list with a new set of values. In the following example, the keyword "Testing" replaces the pre-existing list of values.

**Example:**

```
set keywords to {"Testing"}
```

Using this method, it is also simple to erase a multi-valued list by simply passing in an empty list.

**Example:**

```
set keywords to {}
```

**Searching the Catalog**

To perform a search in a Portfolio catalog, use the **find** command. Searches in Portfolio are executed by passing in a text string which represents the search criteria as they are laid out in the Find dialog in Portfolio. The basic functionality is shown below. See the next section for how to formulate the query variable properly.

**Example:**

```
find matching query1
```

**Building the Query**

Searches in Portfolio are executed by passing in a text string which represents the search criteria as they are laid out in the Find dialog in Portfolio. As in the Portfolio Find dialog, the basic query structure consists of three clauses: the field, the operator, and the value. Each of these clauses is passed in as a textual string, and the tab character separates each clause.

**Example:**

```
set query1 to "Filename" & tab & "starts  
with" & tab & "Abstract"
```

```
find matching query1
```

To build a more complex search, a return character must be used to delimit each line. In addition, each line after the first one needs to begin with the join condition (either **and** or **or**). Below is an example of a two-line search query.

**Example:**

```
set theQuery to "Filename" & tab &
"starts with" & tab & "test" & return &
"and" & tab & "Keywords" & tab & "starts
with" & tab & "key"

set query1 to "Keywords" & tab & "starts
with" & tab & "test"

set query2 to "Width" & tab & "greater
than" & tab & 100

set query3 to query1 & return & "and" &
tab & query2

set query4 to query1 & return & "or" &
tab & query2
```

One exception to this model is when using the **exists** or **does not exist** search operator. In this situation, there is no value being passed in, so the line should only consist of two clauses.

**Example:**

```
set query5 to "Filename" & tab & "exists"
```

# The Portfolio AppleScript Dictionary

## Class advanced cataloging options

Advanced cataloging options for a catalog.

### Properties:

**<Inheritance>** `item [r/o]` -- All of the properties of the superclass.

**skip files without thumbnails** `boolean` -- Whether to skip files lacking thumbnails during cataloging.

**movie thumbnail mode** `poster_frame/time/frame_number` -- The method of generating thumbnails for movies.

**thumbnail size** `size256/size112` -- The pixel size of the thumbnails that are generated.

**frame number** `integer` -- The frame number to use when generating movie thumbnails (for “video thumbnail by frame number” setting).

**movie time** `integer` -- The movie time (in seconds) to use when generating movie thumbnails (for “video thumbnail by movie time” setting).

**extract thumbnail** `boolean` -- Whether to extract thumbnail from source file when possible.

**extract metadata** `boolean` -- Whether to extract advanced metadata during cataloging.

**path keywords** `off/path and volume/file without folder/file and folder name/path without volume`  
-- The method for creating keywords based on file name and relevant folder names.

**index document text** `boolean` -- Extract text from supported documents for use with Find Document Text feature.

## Class application

The Portfolio application object.

### Plural form:

**applications**

### Elements:

**window** by name, by numeric index, before/after another element, as a range of elements, satisfying a test, by ID

**document** by name, by numeric index, before/after another element, as a range of elements, satisfying a test

### Properties:

**<Inheritance>** `application [r/o]` -- All of the properties of the superclass.

## Class cataloging options

Basic cataloging options (either default user options, or saved option presets from the catalog).

### Plural form:

**cataloging options presets**

### Elements:

**field** by name, by numeric index, before/after another element, as a range of elements, satisfying a test, by ID

### Properties:

**<Inheritance>** *item* [r/o] -- All of the properties of the superclass.

**copy or move** *moveValue/copyValue/leaveValue*  
-- Whether to copy, move, or leave in place during cataloging.

**assign properties** *boolean* -- Whether to assign properties (keywords, description) during cataloging.

**rename before cataloging** *boolean* -- Whether files should be renamed before they are cataloged.

**description** *Unicode text* -- The description to apply to cataloged items.

**rename settings** *'PMfr'* -- Settings indicating how files should be renamed during cataloging.

## Class document

A Portfolio catalog.

### Plural form:

**documents**

### Elements:

**gallery** by name, by numeric index, before/after another element, as a range of elements, satisfying a test

**cataloging options** by numeric index, before/after another element, as a range of elements, satisfying a test

### Properties:

**<Inheritance>** *document* [r/o] -- All of the properties of the superclass.

**advanced\_options** *advanced cataloging options*  
-- The document's advanced cataloging options.

**selected gallery** *gallery* -- The currently selected gallery.

**user\_options** *cataloging options* -- The user's default cataloging options.

**current\_options** *cataloging options* -- The currently active cataloging options (may be the user options or one of the presets).

## Class field

A field in a record.

### Plural form:

**fields**

### Properties:

**<Inheritance>** `item [r/o]` -- All of the properties of the superclass.

**custom** `boolean [r/o]` -- If true, field is a custom field.

**name** `Unicode text [r/o]` -- Name of the field.

**ID** `integer [r/o]` -- Unique ID for a custom field.

**value** `Unicode text` -- Value of the field (as a string).

## Class file rename settings:

Settings to apply when renaming files during cataloging.

### Properties:

**<Inheritance>** `item [r/o]` -- All of the properties of the superclass.

**middle text** `Unicode text` -- The text or value used in renaming the middle part of the new name.

**middle mode** `original/textRename/number from/none` -- Rename mode for the middle portion of the new name.

**front text** `Unicode text` -- The text or value used in renaming the front part of the new name.

**end mode** `original/textRename/number from/none` -- Rename mode for the end portion of the new name.

**front mode** `original/textRename/number from/none` -- Rename mode for the front portion of the new name.

**end text** `Unicode text` -- The text or value used in renaming the end of the new name.

## Class gallery

A gallery.

### Plural form:

**galleries**

### Elements:

**record** by name, by numeric index, before/after another element, as a range of elements, satisfying a test, by ID

### Properties:

**<Inheritance>** `item [r/o]` -- All of the properties of the superclass.

**is modified** `boolean [r/o]` -- If the gallery is modified.

**view** `integer` -- Thumbnails, List, or Item view.

**selection** `record [r/o]` -- The selected items.

**name** `Unicode text [r/o]` -- The title of the gallery.

## Class record

An item in a catalog or gallery.

### Plural form:

**records**

### Elements:

**field** by name, by numeric index, before/after another element, as a range of elements, satisfying a test, by ID

### Properties:

**<Inheritance>** *item* [r/o] -- All of the properties of the superclass.

**height** *integer* -- The height (in pixels) of the source file.

**last updated** *date* -- The date the source file was last updated in Portfolio.

**width** *integer* -- The width (in pixels) of the source file.

**number of pages** *integer* -- The number of pages the cataloged item contains.

**filename** *Unicode text* -- The name of the source file.

**file volume** *Unicode text* -- Name of the volume where the source file is stored.

**file type mac** *Unicode text* -- The four-character Macintosh file type of the source file.

**category** *list* -- The list of category IDs assigned to the record.

**created** *date* -- The date the source file was created.

**name** *Unicode text* [r/o] -- The name of the record (same as the file name).

**vertical resolution** *real* -- The vertical resolution (in DPI) of the source file.

**watermark url** *Unicode text* -- The watermark url.

**file size** *integer* -- The size of the file (in kilobytes).

**short file name win** *Unicode text* -- The 16-bit (DOS) file name of the source file.

**placeholder** *integer* -- 1 if the record is a placeholder record and zero if not.

**zone mac** *Unicode text* -- The name of the AppleTalk zone in which the file is located.

**color mode** *integer* -- The color mode of the source file, stored as an integer.

**description** *Unicode text* -- The record's description.

**horizontal resolution** *real* -- The horizontal resolution (in DPI) of the source file.

**watermarked** *integer* -- 0 - Digimarc ID Unknown, 1 - Digimarc ID not detected, 2 - Digimarc ID detected.

**file path** *Unicode text* -- The full path to the source file.

**cataloged** *date* -- The date the record was created (source file was cataloged) in Portfolio.

**ID** *integer* [*r/o*] -- Unique ID of a record.

**keywords** *list* -- A list of keywords for the record.

**last modified** *date* -- The date the source file was modified.

**thumbnailsize** *integer* -- The size of the thumbnail stored in the record (either 0, 32, 64, 112, or 256).

**thumbnail** *anything* -- The file's thumbnail image.

**creator mac** *Unicode text* -- The four-character creator code of the source file.

**extension** *Unicode text* -- The extension (suffix) of the filename.

**directory path** *Unicode text* [*r/o*] -- Full path to the parent directory of the source file.

## catalog (application)

Tell Portfolio to catalog a file.

**catalog** *Unicode text* -- The file or folder to be cataloged.

[**action** *nothing/select\_catalog/new\_catalog/open\_local/open\_served/open\_sql*] -- An additional action code (optional).

[**to gallery** *anything*] -- The gallery to which the items should be added. If omitted, items are added to the frontmost gallery window.

[**to catalog** *Unicode text*] -- The path to the catalog file to which the items should be added. If omitted, items are added to the frontmost gallery window.

[**show options** *boolean*] -- Whether to show cataloging options dialog during cataloging.

**Result:**

'ctlg' -- the reply for the command

## collect: (gallery)

Collect and back up the records in a gallery.

**collect** *reference* -- the object for the command

[**preserve file hierarchy** *boolean*] -- Whether to preserve the file hierarchy in the collection. The default setting is no.

[**selected items only** *boolean*] -- Whether to collect only the currently selected items. The default setting is no.

[**using preview mode** *size 480/size 256/original size/size 1024/size 640/use originals*] -- Which preview mode to use. The default setting is original size.

**to location** *Unicode text* -- The destination directory for the collected files. Must be an extant directory.

**[create mac browser** *boolean*] -- Whether to create a Mac browser app. The default setting is no.

**[create archive catalog** *Unicode text*] -- Path and filename of the catalog to create. If this parameter is not used, a catalog will not be created.

**[create windows browser** *boolean*] -- Whether to create a Windows browser app. The default setting is no.

## create web pages (document)

Create web page(s) of the current selection of the frontmost gallery.

**create web pages**

## debug test

Prints a statement to the log or console showing that this command was received.

**debug test** *reference* -- the object for the command

## display (gallery, record)

Tell an object to display itself.

**display** *reference* -- the object for the command

## export field values: (gallery)

Export field values from a Portfolio Database to a Text File.

**export field values** *reference* -- the object for the command

**[using template** *list*] -- The names of record fields to be exported.

**toFile** *Unicode text* -- The destination (including file name) for the text file. If the file already exists, it will be overwritten.

## find (document)

Conduct a search of a Portfolio catalog.

**find** *reference* -- the object for the command

**matching** *list* -- An array of search criteria.

## gallery copy (application)

Copy records from one gallery of a catalog to another.

**gallery copy** *anything* -- The set of records to copy.

**to** *gallery* -- The destination gallery for the copy.

## import field values (gallery)

Import field values from a text file into a Portfolio catalog.

**import** *reference* -- the object for the command

**from** *Unicode text* -- The path of the file to import.

[**Using** *Unicode text*] -- The name of the saved import set to use.

## is cataloging (application)

Check whether Portfolio is currently cataloging files.

**is cataloging** *reference* -- the object for the command

### Result:

'ISCA' -- the reply for the command

## make catalog (application)

Create a new Portfolio catalog.

### make catalog

**with previews** *boolean* -- Whether the new catalog should generate screen-resolution previews for images.

## make custom field: (document)

Makes a new Custom Field. Requires administration access level. Will overwrite a custom field of the same name.

**make custom field** *reference* -- the object for the command

[**restrict to predefined list** *boolean*] -- Whether or not to restrict the field's values to the values of the predefined list. Only defined for fields with predefined lists.

**with name** *Unicode text* -- Name for the custom field.

[**allow multiple entries** *boolean*] -- Whether or not to allow multiple entries. Not defined for fields of WebURLType and TextBlockType.

[**precision** *integer*] -- The precision of a decimal field. Legal values are 0 to 8. Only defined for fields of type DecimalType.

[**use predefined list** *list*] -- A list of strings to use as a predefined list for the field.

[**max chars** *integer*] -- The max size of a text field.

[**display time** *boolean*] -- Whether or not to display the time for a date. Only defined for fields of DateTimeType.

**of type** *TextType/WebURLType/DateTimeType/NumberType/DecimalType/TextBlockType* -- Which type.

## make gallery (document)

Create a new gallery.

**make gallery** *reference* -- the object for the command

**with name** *Unicode text* -- Name of new gallery

## make record (document)

Create a new, empty record in the front Gallery.

**make record** *reference* -- the object for the command

**with name** *Unicode text* -- The name to assign to the new record.

### Result:

'ReID' -- the reply for the command

## open catalog (application)

Open a catalog.

**open catalog** *Unicode text* -- The path to the catalog.

**[using mode** *browser/editor/publisher/reader/administrator*] -- The access mode to use when opening the catalog.

**[with username** *Unicode text*] -- The user's name for the catalog.

**[with password** *Unicode text*] -- The user's password for the catalog.

## open server (application)

Open a Portfolio catalog residing on a Portfolio Server.

**open server** *Unicode text* -- The server address, in the form of "portfolio://ip-address/catalog-name" for regular served catalogs, or "portfoliosql://ip-address/catalog-name" from SQL-served catalogs.

**[using mode** *publisher/editor/reader*] -- The access mode to use when opening the catalog. If not supplied, Reader Mode is used.

**[with username** *Unicode text*] -- The username for access to the served catalog.

**[with password** *Unicode text*] -- The user's password for access to the served catalog.

## remove selection (gallery)

Remove selected items from the gallery.

**remove selection** *reference* -- the object for the command

**[from catalog** *boolean*] -- Removes record from catalog as well as gallery

## select (anything)

Select one or more objects.

**select** *anything* -- The object(s) to select.

## serve catalog (application)

Serve a catalog.

**serve catalog** *reference* -- the object for the command

**ip address** *Unicode text* -- The IP address for the catalog to serve.

[**autoserve** *integer*] -- Whether to always serve this catalog.

**using name** *Unicode text* -- The partial path of the catalog to serve.

## serve sqlcatalog (application)

Serve a SQL catalog.

**serve sqlcatalog** *reference* -- the object for the command

[**password** *Unicode text*] -- The user's password.

**ip address** *Unicode text* -- The IP address for the catalog to serve.

[**autoserve** *integer*] -- Whether to always serve this catalog.

**using name** *Unicode text* -- The partial path of the catalog to serve.

**user** *Unicode text* -- The user name for the connection.

**sql server name** *Unicode text* -- The SQL server name.

## sort (gallery)

Sort records in the gallery view.

**sort** *reference* -- the object for the command

**by** *Unicode text* -- The name of the field to sort by.

[**order** *descending/ascending*] -- Which order to sort.

## unselect all (gallery)

Unselect all records.

**unselect all** *reference* -- the object for the command

## unserve catalog (application)

Stop serving a catalog.

**unserve catalog** *reference* -- the object for the command

**ip address** *Unicode text* -- The IP address for the catalog to unserve.

**using name** *Unicode text* -- The partial path of the catalog to unserve.

**timeout** *integer* -- The number of seconds to wait before forcing users off.



