Extensis Portfolio 4.0

AppleScript Reference

Table of Contents

Overview

The purpose of this document

This document is meant as a brief overview to the Portfolio commands available via the scripting interface in Extensis Portfolio 4.0 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. This document assumes a basic level of familiarity with AppleScript.

*What this document is **not***

This document is meant neither as an introduction to AppleScript nor as a tutorial for how to script Portfolio to perform specific tasks.

Plenty of information is already available on how to use AppleScript. An excellent starting point for AppleScript information is available on the Internet at http://www.apple.com/applescript. This site contains a great deal of useful information on AppleScript as well as links to other scripting-related web sites.

For more information on how to script Extensis Portfolio, refer to the additional files provided on the Extensis CD. Additional information is also available on Extensis' website at http://www.extensis.com in the Portfolio product section.

Starting Portfolio

As with any application in AppleScript, use the tell command to address Portfolio. If Portfolio is not active, the application will launch. Use the activate command to bring the application to the front, if desired

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
```

Working with Catalogs

Opening catalogs

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

Example:

```
open alias "Macintosh HD:Port4: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.

Example:

```
open alias "Macintosh HD:Port4: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 "192.0.0.0/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.

Example:

```
open server "192.0.0.0/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.

Example:

```
new catalog file "Macintosh HD:Port4:Test.fdb"
```

Working with Galleries

Once a catalog has been opened or created, the majority of commands will be directed at the Gallery object (in fact, there is no Catalog object in Portfolio's script model).

Selecting a Gallery

The majority of 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. In many situations, scripts will simply manipulate 'the front gallery' (the active gallery from the user's perspective). However, galleries can also be referenced by index or by name, as well.

Examples:
of front gallery *-- Currently selected gallery*
of gallery 5 *-- Gallery with an index of 5*
of gallery "test.fdb" – *Gallery titled 'test.fdb'*

Note: 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 used indexed, single-valued fields for sorting.

Example:
sort **front** gallery by "Filename"

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:Images:" to front gallery
```

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:

```
if (is cataloging) is true then
	-- Wait here until it is done
end if
```

Cataloging Options

Several of the options in the Cataloging Options dialog have been exposed to the scripting interface. In particular, the Modification Method, Thumbnail Size, and Thumbnail Quality 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 cataloging options of front gallery to {modify method:add, thumbnail size:size256,
	thumbnail quality:low}

set modify method of cataloging options of front gallery to addunconditionally
```

It is generally recommended to assign the original set of properties to a variable so that they can be reset after the script has completed its cataloging process(es).

Example:

```
set pOptions to properties of cataloging options of front gallery
-- Change the properties, do some cataloging actions, etc.
set properties of cataloging options of front gallery to pOptions --Restore the options to their original
	state
```

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) or by ID. 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:
record 1 -- *Reference by index*
record id 10 – *Reference by ID*

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 every** record **of** selection **of front** gallery **as** list

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:
select records 10 **thru** 13 **of front** gallery
select record id 68 **of front** gallery
select **every** record **of front** gallery

Clearing the selection

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

Example:
unselect all record **of front** gallery

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 description of record 1 of front gallery
```

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:
```
get field “description” of record 1 of front gallery
get field 25 of record 1 of front gallery
```

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 the first record of the front gallery
```

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 field "Description" of record 1 of front gallery to "This is a test description"
```

Notes on Setting Field Values

Data Types

For scripting purposes, all values for setting 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. The reason for this is that Portfolio uses its own internal validation routines to determine whether the data being passed in is appropriate for the field type. The following example shows the “Last Updated” field (a date field) being set by passing in a string.

Example:

```
set last updated of record 1 of front gallery to "Monday, December 1, 1997 12:00:00 AM"
```

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 last updated of record 1 of front gallery 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 of record 1 of front gallery 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 (so that the keyword list now only consists of “Testing”).

Example:

```
set Keywords of record 1 of front gallery 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 of record 1 of front gallery 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 on how to formulate the query variable properly.

Example:

```
find record of front gallery matching theQuery
```

To find all the records in the catalog, simply pass in an empty query string.

Example:

```
find record of front gallery matching ""
```

The scope of the search can also be defined with the command by appending the optional '`in all records`' or '`in shown records`' clause. These two options correspond to the 'Search in Gallery' checkbox in the Portfolio Find dialog. By default, the scope is all records in the catalog.

Example:

```
find record of front gallery matching theQuery  in all records
find record of front gallery matching theQuery  in shown records
```

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 theQuery to "Keywords" & tab & "starts with" & tab & "test"
find record of front gallery matching theQuery in all records
```

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"
```

Note that 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 theQuery to "Filename" & tab & "exists"
```

Importing and Exporting

The Portfolio scripting interface also provides for importing and exporting data.

Exporting HTML

The scripting interface for the Export HTML command is very similar to the Export HTML command within Portfolio. To export the current selection using the layout of the gallery as the template, use the `export html` command. The file being passed in is the full path to the location of the first HTML file to be generated. As in Portfolio, the Export HTML command is selection-based, so a record set must be selected for the command to work correctly.

Example:

```
export html file "Macintosh HD:Port4:test.html"
```

Note: Any alerts that might come up during the Export process (such as overwrite warnings) are automatically overridden. If files of the same name already exist at the location specified, they will be automatically overwritten.

To use a pre-existing HTML template append the optional `using template` clause.

Example:

```
export html file "Macintosh HD:Port4:test.html" using template "html1"
```

Note: The template specified must already exist in the selected catalog; it cannot be created through the scripting interface.

Importing text data

To import text data using Portfolio's Import Field Values command, use the `import` command. The file passed in is the text file to be used, and the `using` clause specifies the Import Saved Set to be used for matching up the incoming data with Portfolio fields.

Example:

```
import file "Macintosh HD:Test:test data.txt" using "import settings"
```

Note: The template specified must already exist in the selected catalog; it cannot be created through the scripting interface.