Outline Sharing 2.0

UserLand Software, Inc.

UserLand Software is located at 400 Seaport Court, Redwood City, CA 94063. 415-369-6600, 415-369-6618 (fax).

Email: userland.dts@applelink.apple.com. If you're an AppleLink user, check out the UserLand Discussion Board under the Third Parties icon. CompuServe users enter GO USERLAND at any ! prompt.

Comments, questions and suggestions are welcome!

Background

Outlines are very useful structures because they allow interactive editing of hierarchies. Hierarchies come up all the time in application software and operating systems. The Macintosh file system, with its folders, sub-folders and sub-sub-folders is a hierarchy. Outlining is an important component of word processing and presentation software. And as Frontier has shown, scripts and program source code can be elegantly edited as an outline based hierarchy.

We developed the Outline Sharing Toolkit to connect several applications we're working on at UserLand Software. We'd like other developers to connect their hierarchy processors to the outline sharing protocol because we believe it will open up some very interesting opportunities for script writers.

Outline Sharing is fully compatible with the other components of Frontier SDK 1.1. You could include the Toolkit in a UCMD, or wire up a system event handler to an Apple Event using the IAC Tools library.

The OutlineSharer Demo App

We provide a small application called OutlineSharer that uses the Outline Sharing Toolkit to implement several very simple outline operations. It's provided mainly as a way of illustrating the use of the Toolkit.

It implements four outline-related Apple Events. A Frontier install file is provided for OutlineSharer. In the OutlineSharer.examples table there are several scripts which test out the Apple Events implemented in OutlineSharer.

Overview of the Toolkit

outlinesharing.c

The Outline Sharing Toolkit is implemented in a single C source file, outlinesharing.c.

Three routines are included that make it possible to send and receive outlines over the Apple Event channel as easily as you would send a number or a string.

IACgetoutlineparam lets you extract an outline parameter from an Apple Event message. Use IACpushoutlineparam to call other programs with an outline parameter. To return an outline as the result of an Apple Event message, call IACreturnoutline. These routines follow the same calling conventions as routines implemented in iac.c, included in this package.

opPack turns an in-memory outline into a contiguous handle that you can save in your program's file format or in the resource fork of a data file, or your program. opUnpack turns a packed outline into an in-memory outline structure.

Over 30 other routines are provided to traverse an outline, get or set the text of any headline, find out how many lines there are, build outlines and restructure outlines. We provide documentation for routines that are central to the transport of outlines over the Apple Event channel, but rely on the source code and comments to document the routines that deal with in-memory outline structures.

outlinerecords

Outlines are represented in memory by a heap-allocated structure called an outlinerecord. All routines operate on a global outlinerecord called outlinedata. You can keep a lot of outlines around, just be sure that before you call one of the Outline Sharing Toolkit routines, that the outlinedata global points to the one you want worked on. Two routines, opPushOutline and opPopOutline, are provided so you can set the global without disturbing any other operations which may be in progress.

Each outlinerecord contains a handle that links to the first summit in the outline structure. This handle is called hsummit. It points to a structure of headrecords, described in the following section.

Outlinerecords have an outlinerefcon field, which is left entirely to the implementer. You can use this field to attach additional structures to the outlinerecord data type.

headrecords

Each node in an outline is represented by a headrecord. Each headrecord has a string handle containing the text of the headline. It also has four handles pointing to adjacent nodes: headlinkdown points to the next headline at the same level (the next “sibling”), headlinkup points to the previous sibling, headlinkright points to the headline’s first child, headlinkleft points to the headline’s parent. If a headline has no children, its headlinkright handle points back at the node itself. The same is true for all other handles. In general, if there’s no place to go, the handle points back at the node itself.

Every headrecord has a boolean indicating whether or not it’s expanded. If one headline at a level has its expanded bit set, all headlines at that level must also have their expanded bits set.

In outlinesharing.c headrecords are also referred to as nodes.

Like outlinerecords, headrecords also have a refcon field, allowing you to maintain links into other data structures using headrecords as a table of contents or an index.

Outline Sharing Routines

Creating and disposing outlinerecords

pascal Boolean opNewOutlineRecord (hdloutlinerecord *);

Creates a new outlinerecord with a single summit with an empty headline. Returns true if it worked, false otherwise. Has no effect on the outlinedata global.

pascal void opDisposeOutlineRecord (hdloutlinerecord);

Disposes of all memory used by the indicated outlinerecord, including the refcon handle linked into the outlinerecord and linked into each headrecord.

For an example, see setoutlineverb in main.c.

Setting the current outline

pascal Boolean opSetTarget (hdloutlinerecord);

Sets the current outline handle, outlinedata, to point at the indicated outlinerecord.

pascal Boolean opPushOutline (hdloutlinerecord);

Sets the current outline handle, outlinedata, to point at the indicated outlinerecord. Before setting the global, it pushes its current value on a stack so it can be restored with opPopOutline.

pascal Boolean opPopOutline (void);

Restores the outlinedata global to the value it had before the last call to opPushOutline.

Sending and receiving outline structures thru Apple Events

pascal Boolean IACgetoutlineparam (OSType, hdloutlinerecord *);

Used in an Apple Event handler to get an outlinerecord parameter from the current event. The OSType value is the keyword for the parameter, for example '----'.

If there is no parameter with the indicated keyword, or it’s the wrong type (not an outline), or there was a failure in unpacking the outline, an error is automatically reported to the Apple Event caller. Otherwise, a handle to the outlinerecord is returned in the handle passed as the second parameter.

For an example, see setoutlineverb in main.c.

pascal Boolean IACpushoutlineparam (hdloutlinerecord, OSType);

Packs an outline into a single contiguous handle and pushes it on the current Apple Event. Call this routine when you're sending an event to another application that takes an outline as a parameter.

Returns false if there was an error packing the outline or in pushing it on the Apple Event record.

pascal Boolean IACreturnoutline (hdloutlinerecord);

Used in an Apple Event handler to return an outline as the result of the Apple Event.

Returns false if there was an error packing the outline or in pushing it on the Apple Event record.

For an example, see getoutlineverb in main.c.

Packing/unpacking outlines into/from contiguous handles

pascal Boolean opPack (Handle *);

Packs the current outline into a contiguous handle. Returns false if it ran out of memory while packing the outline.

This routine allocates a new handle if the handle passed to it is nil on entry. If it's not nil, it appends the packed outline to the handle passed to it. This allows you to call opPack to save an outline that's part of a more complex data structure.

pascal Boolean opUnpack (Handle, long *);

Unpacks the contiguous handle into the current outline, pointed to by the outlinedata global. Returns false if it ran out of memory while unpacking the outline.

If the handle contains only an outline, set the long to 0 before calling opUnpack. However, if the outline is only part of the handle, set the long to the offset that the packed outline begins at. When opUnpack returns, the long will contain the offset of the first byte in the handle after the outline.

Navigating thru an outline structure

pascal Boolean opGo (hdlheadrecord *, opDirection);

Starting at the indicated headrecord, move in the indicated direction. Returns false if there is no way to navigate in the direction.

For example, if the direction is up (constant: opUp) the headrecord returned is the previous sibling of the starting headrecord. opGo would return false if you try to navigate up from the first headline at a given level.

Getting and setting headline strings

pascal void opGetHeadString (hdlheadrecord, Str255);

Returns the text linked into the indicated headrecord.

pascal Boolean opSetHeadString (hdlheadrecord, Str255);

Sets the text of the indicated headrecord. Returns false, without modifying the existing text, if it was unable to allocate a new handle to hold the text.

Traversing the outline

pascal Boolean opVisitOutline (opvisitcallback visit);

Traverses the entire outlinerecord pointed to by outlinedata, calling the visit routine for each headrecord in the outline.

The callback routine must take a single parameter, a hdlheadrecord, and return a boolean. If the visit routine returns false, the traversal halts immediately. This can be used to implement a search, or can indicate an error. If the visit routine returns false, opVisitOutline returns false to its caller. If all calls to the visit routine returns true, opVisitOutline returns true.

For an example, see uppercaseoutlineverb in main.c.

Other, more exotic traversal routines are also provided in outlinesharing.c.