DocAssembler

An application to collect
and structure text documents
on the Macintosh

Version 1.0 - March 1996

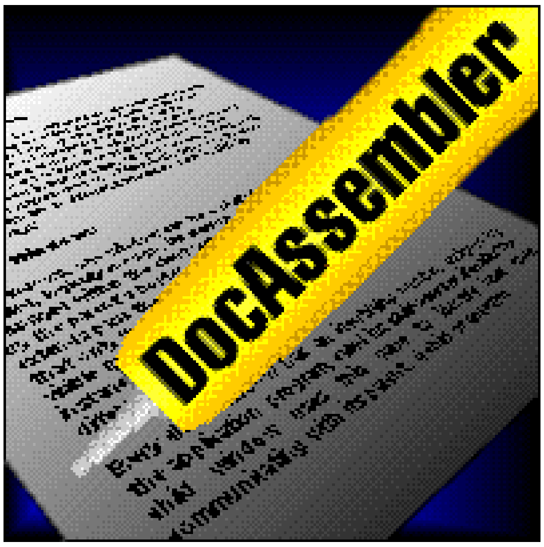


Table of contents

What is DocAssembler?

DocAssembler is an application that collects text documents, which are supposed to be "sections" of a global document and produces a single target document that comprises them all. Between these two phases, the user can arrange the structure of the sections, by changing both their order and their hierarchical level, relying upon DocAssembler for automatic correct indexing.

You might find this application useful if you need to assemble a manual or other forms of documentation, especially if you work in a collaborative environment where different people is responsible for different parts. You could even put the parts togheter automatically using DocAssembler's scripting features.

Or you may use it if you need to repeatedly produce text documents whose content does not differ much. Such documents can be contracts or tenders, which usually have standard parts that remain unchanged in every situation, while other small portions change accordingly to the specific context.

Both the source documents and the output document can be specified in a variety of formats, thanks to the Claris XTND technology support. DocAssembler will read and write any text document for which an XTND translator is available. This is the same technology used by Claris as well by a number of commercial and shareware products.

If the Drag and Drop extension is available, DocAssembler lets you collect the source files simply by dragging them from the Finder or change their order dragging them around the document window.

DocAssembler supports all the standard Apple Event suites, and is both scriptable and recordable. Besides, it defines a few custom events so that you can completely remote control it using AppleScript. To learn more about this subject, see the "AppleScript reference" chapter and the sample scripts that come with the package.

System requirements

DocAssembler requires a Macintosh with:

• MC68000 processor or better
• 4 Mb RAM
• Color QuickDraw
• Mac OS 7.1 or later
• Claris XTND System

• Drag and Drop extension (optional, but strongly recommended)
• AppleScript (optional)

The minimum required amount of memory is 700 Kb and should be adequate for simple and medium tasks using text only documents. Complex documents containing pictures will require more generosity.

Shareware information

This software is © 1996 Amedeo Farello, all rights reserved.

Amedeo Farello
Loc. Vereytaz, 7
11018 Villeneuve (AO)
ITALY

e-mail: <farello@mbox.vol.it> or <farello@kagi.com>

This software is distributed as SHAREWARE. This means that after a 15 days trial period, if you decide to continue using it, you are required to register your copy by sending the requested fee to the author.

Of course, if you don't do that and you still use the software, it is unlikely that you will ever face a trial. What is likely instead, is that I will stop its development. So, if you find it useful, it could be a nice idea to register.

Remember that shareware is only an alternative distribution method. It simply lets you try the product before you buy. Please, honor this concept and register your copy.

Limitations to unregistered copies

Each time you launch an unregistered copy of DocAssembler, a startup screen is shown recalling the principles of shareware and you can’t continue for at least 10 seconds. Besides, unregistered copies won’t accept more than 5 source files in their documents. These limitations won’t prevent you from evaluating this software.

If you register your copy, you will receive a code that fully enables it and removes that annoying startup screen (and your sense of guilt). Registration gives you the legal right to use this application and its possible next versions.

Available licenses

When you register, you can ask for 3 different types of license: “Single User Licenses”, “Site Licenses” or a “World-Wide License”.

A “Single User License” allows the use of the registered copy by any number of people, as long as there is no possibility of it being used simultaneously on more than one machine.

A “Site License” covers all locations for your organization within a 160 kilometer radius of your site (100 miles). One big advantage of a “Site License” is that you do not need to keep track of how many people at your site are using the software.

A “World-Wide License” covers all locations for your organization on the planet earth.

How to register

Paying for my software is fairly simple. Open the **Register** program that accompanies my software. Enter your name, your e-mail or postal address, and the type and number of licenses you desire for each program you wish to purchase. Save or Copy or Print the data from the Register program and send the data and payment to Kagi Shareware. Kagi Shareware handles my payment processing.

If paying with **Credit Card** or **First Virtual**, you can e-mail or fax the data to Kagi Shareware. Their e-mail address is <shareware@kagi.com> and their fax number is +1 510 652-6589. You can either Copy the data from Register and paste into the body of an e-mail message or you can Save the data to a file and you can attach that file to an e-mail message. There is no need to compress the data file, it's already pretty small. If you have a fax modem, just Print the data to the Kagi fax number.

Payments sent via e-mail are processed within 3 to 4 days. You will receive an e-mail acknowledgement when it is processed. Payments sent via fax take up to 10 days and if you provide a correct internet e-mail address you will receive an e-mail acknowledgement.

If you are paying with **Cash** or **USD Check** you should print the data using the Register application and send it to the address shown on the form, which is:

Kagi Shareware
1442-A Walnut Street #392-AF
Berkeley, California 94709-1405
USA

You can pay with a wide variety of cash from different countries but at present if you pay via check, it must be a check drawn in US Dollars. Kagi Shareware cannot accept checks in other currencies, the conversion rate for non-USD checks is around USD 15 per check and that is just not practical.

If you have a purchasing department, you can enter all the data into the Register program and then select Invoice as your payment method. Print three copies of the form and send it to your accounts payable people. You might want to highlight the line that mentions that they must include a copy of the form with their payment.

Kagi Shareware can not invoice your company, you need to act on my behalf and generate the invoice and handle all the paperwork on your end.

Please do not fax or e-mail payment forms that indicate Cash, Check or Invoice as the payment method. As far as we know, there is still no technology to transfer physical objects via fax or e-mail and without the payment, the form cannot be processed.

Payments sent via postal mail take time to reach Kagi Shareware and then up to 10 days for processing. Again, if you include a correct e-mail address, you will hear from Kagi Shareware when the form is processed.

About the registration code

If you entered an e-mail address in the registration form, I will use it to send you the code as soon as I am informed of your payment. If you do not have e-mail, I will have to send you a letter: please understand that this will take more time.

I want your feedback!

I would like to know what you think about my work, so, even if you are not going to register, let me know your impressions, suggestments and bugs discoverings.

Distribution restrictions

This software can be freely distributed as long as it is not modified, the original package is included in its entirety and there's no charge for it. It may not be included in any commercial package without my written consent.

All online services and bulletin boards may make it available to their users at no charge other than the normal connection fees.

All non-profit user groups may distribute it at no charge.

All magazines may publish it on floppy disk without asking me first, as long as I get a copy of the issue containing my software.

All CD-ROM shareware collections and CD-ROM magazines may include it without my prior consent, as long as I get a copy of the CD-ROM.

Software license

About the author

Although I have a degree in Architecture, many years ago I discovered that my real passion was to tinker with computers, so I work full time as a software developer now. I started programming on an Apple II in 1982, then switched to MS-DOS in 1985 and finally to Macintosh in 1990.

I currently use the C++ and C languages for all my development projects, but I have some little experience with Pascal and Fortran as well.

I am very interested in Computer Graphics, information storage and rendering methods, human-computer interaction problems and everything related to the software development process in the design and communication fields. Of course, all is Internet related also interests me.

In the past I have written software dealing with:

- geometric modeling (2D-3D curves and surfaces, boolean operations between polygons)
- user interface ("intelligent" support to technical drafting)
- rendering (using flat shade, Gouraud, Phong, ray-tracing and radiosity techniques)
- software localization (resources-to-glossary-to-resources)
- CAD systems integration (parametric generation of models)
- programming utilities (source code collection and formatting)
- structural engineering (linear and non-linear analysis of plane frames and sections, linear analysis of 3D truss structures)

Note: I am currently considering job offers as a Macintosh programmer.

Acknowledgements

I wish to thank Alessandro Levi Montalcini for his intelligence and generosity.

Using DocAssembler

Please be aware that this application cannot function unless the Claris XTND System is installed on your machine. The XTND software is commonly distributed along with many, if not all, Claris software products and many others. Given the XTND System is available and installed, you don't need to do any particular operation in order to use this program, I just suggest you to copy it onto your hard disk.

When you start DocAssembler, a new empty document is created. A document consists in a list of file references that you have to fill. You can build the list in various ways, the easiest is to drag and drop file icons directly from the Finder or file references from another DocAssembler document.

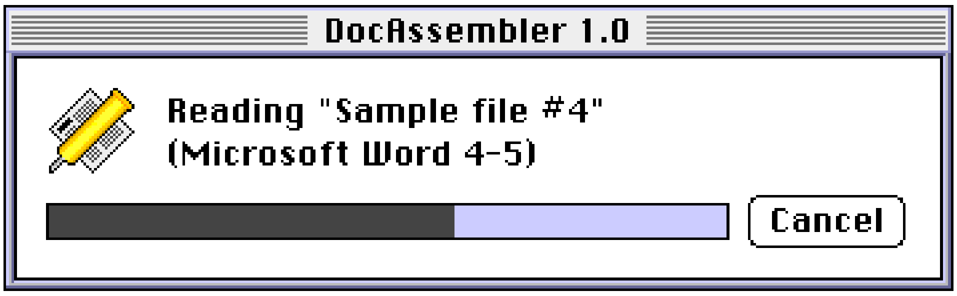
New files are always appended at the end of the list, you can change their order lately using drag and drop. If you try to add a file that's already in the list, you are warned by the program and the operation is canceled. Each file reference is shown with its type and creator, and three fields in its record show if the file is really available (since you can add a file and then delete it), and if a proper XTND translator and the creator application are available. If the application with which you created the file is available, double clicking on the file record will ask it to edit the file.

You can assign each file a "hierarchical" level between the four available. By default, new files receive the topmost one. You can change the level using the commands from the "Document" menu.

When the list is completed, you choose the "Produce Document..." command from the "File" menu, which also has a button equivalent in the toolbar, the target document production process begins.

First, all the source documents archived in the current DocAssembler document are read and their data copied to a temporary file created in the same folder of the application. This phase is shown by a progress indicator along with the name and the format of each file being read.

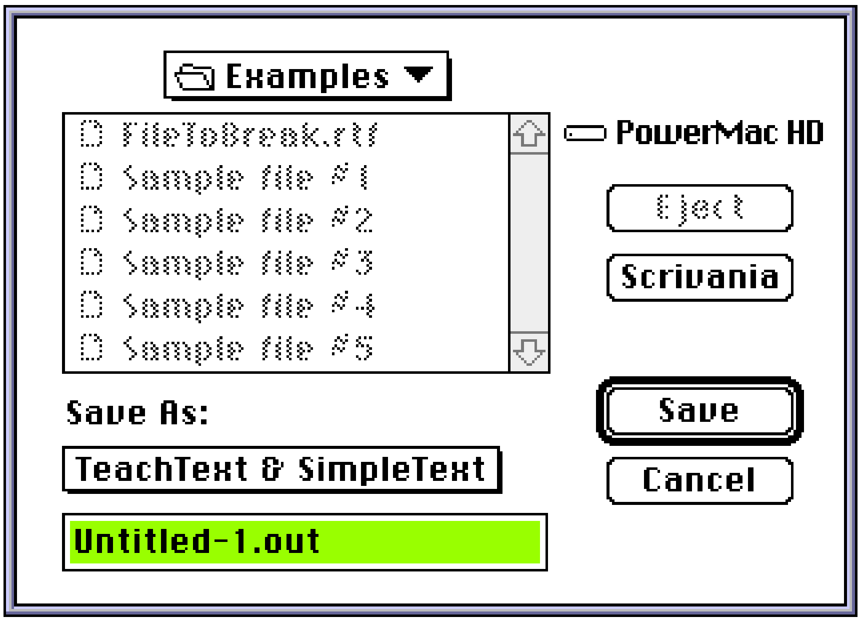
The input progress indicator


DocAssembler 1.0

Reading "Sample file #4"
(Microsoft Word 4-5)

Then, a dialog will be presented to let you choose the name and the format of the output document. The popup menu at the bottom of the dialog shows the list of the available translators.

The dialog for choosing the output document

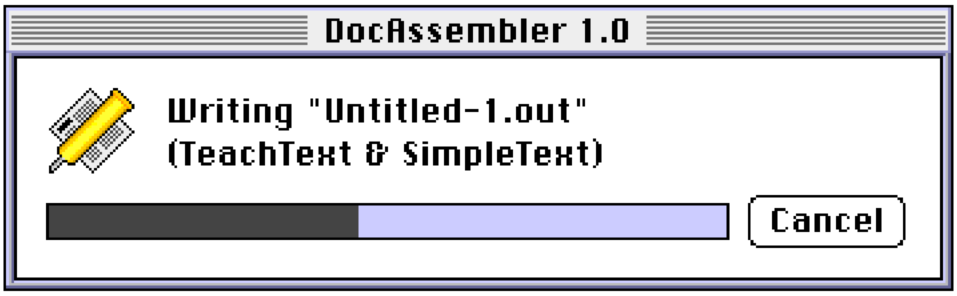


TeachText & SimpleText

Untitled-1.out

Finally, the output phase will begin, again shown by a progress indicator. At this stage, a window shows the name of the file being written and the translator in use. You can stop the process at any time by pressing the “Cancel” button.

The output progress indicator

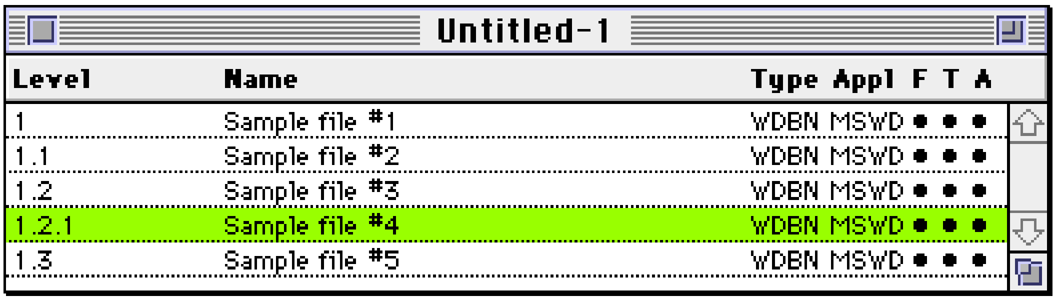


Writing "Untitled-1.out"
(TeachText & SimpleText)

Anatomy of a DocAssembler window

You can open multiple documents in DocAssembler, their number being limited only by the available memory. Each document has a single window, arranged as a list of records, each one of them refferring to a different source file. The fields are structured as described below.

A DocAssembler document window



Untitled-1

Level	Name	Type	Appl	F	T	A
1	Sample file #1	WDBN	MSWD	●	●	●
1.1	Sample file #2	WDBN	MSWD	●	●	●
1.2	Sample file #3	WDBN	MSWD	●	●	●
1.2.1	Sample file #4	WDBN	MSWD	●	●	●
1.3	Sample file #5	WDBN	MSWD	●	●	●

The "Level" field

This field contains the index that DocAssembler automatically assignes to the source file on the basis of its current “hierarchical” level and position in the list. This index can be included in the output document just in front of the very first text of each section, thus allowing you to structure the document, relying upon DocAssembler for a correct indexing scheme.

To learn more about levels, see the "How to structure the document" section.

The "Name" field

This field shows the name of the source file.

The "Type" and "Appl" fields

This fields show the "file type" and the "file creator" of the source file. As you probably know, these informations are used by the System to assign an icon to each file and to open the correct application when you double click on it. Here they are used to provide concise information about what kind of file are you dealing with.

During the development of DocAssembler, I started by using the full creator application name returned by the desktop database to identify the source files (much in the same way used by the "Get Info" command of the Finder), but I soon realized that this was wasting a great amount of screen space, so I turned to this compact method.

The "F" field

This field shows if the source file is available, that is, if DocAssembler was able to find it (in this case a "•" is shown) or not (in this case a "-" is shown). Of course, when you first add a source file to a document, it is always available, but later the file could be deleted or moved in a way that the Alias Manager is not able to resolve.

In order to produce the output document, every source file has to be available, otherwise you will get an error message. When this happens, you can choose between removing its record from the list or try to restore its original position.

The "T" field

This field shows if a proper XTND translator is available for the source file. In this case a "•" is shown, otherwise a "-" is shown.

DocAssembler lets you add any kind of file to a document, presuming that you know what you are doing and that if a translator isn't available now, maybe it will in the future.

Due to how the XTND System is conceived, to verify if a particular translator matches a file, a real File System Specification record must be provided instead of just passing a file type descriptor. So, if a real file isn't available, the XTND System will fail to confirm its translatability, even if it should. This is why an unavailable file will never have the "T" field set.

Anyway, when you try to produce the output document, every source file has to have its proper translator available, otherwise you will get an error message.

The "A" field

This field shows if the application that created the source file is available. In this case a "•" is shown, otherwise a "-" is shown. When the application is available, a double click on the record will open it along with the source file.

Since this condition is assumed from the "file creator" information beared by the source file, there could be situations when even if the creator application isn't available, the file could be opened by others (for example if the file is an ordinary 'TEXT' file). In this case, if you want to be able to edit the source file from DocAssembler, you will have to change its "file creator". (There are many shareware/freeware utilities that can do this).

The availability of the creator application does not interfere in any way with the production of the output document.

When is the list updated?

The list is updated each time a source file is added or removed, or when its level or position is changed.
Besides, the list is updated when the "Produce Document" command from the "File" menu is issued. Finally, you can explicity request the list updating with the "Rebuild List" command of the "Document" menu.

How to collect the source files

DocAssembler provides three different methods for collecting the source documents: using the standard “Open” dialog, using Drag and Drop and using AppleScript.

Accept / reject criteria

Although these methods work quite differently, they share a common behaviour: when you tell DocAssembler to get a new file, it tries to understand if the file is manageable (I am sure you would not like it to accept an application or a control panel!).

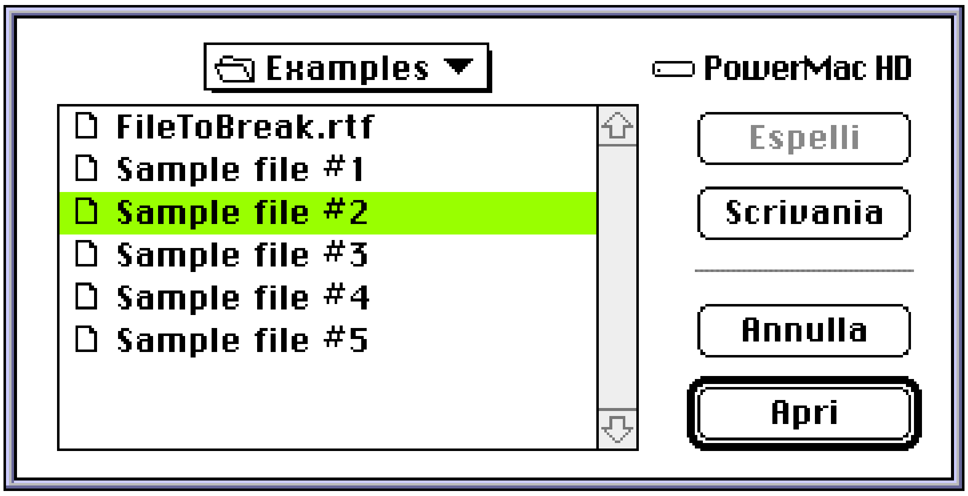
Since DocAssembler tries to overcome file format limitations, it can't simply recognize files from a finite set of file types. So, its strategy is to accept any file except those it can recognize as not compatible for sure, like applications, system files, PICT files, etc.

While this kind of approach lets you collect files produced with a photo retouching application or a three-dimensional rendering program, for which a text import translator is not not likely to exist, it will not limit your horizons: if you want, you can write the next major word processing application along with a proper XTND translator and still use DocAssembler to process your files.

Don't worry about aliases: every time you add one, it will be automatically resolved and the real file, if found, will be added instead.

The "Open dialog" method

This method uses the familiar “Open” dialog to let you select a file to add to the current active document. To use it, select the “Add File...” command from the “Document” menu or push the equivalent button in the toolbar and choose the file.



In the usual list you will see only the files DocAssembler can accept (refer to the "Accept / reject criteria" paragraph).

This is probably the most reassuring way to collect files, but it is also the slowest, since it works with only one file at the time. The real reason for it is there, is to let those users who don't have neither the Drag and Drop extension nor AppleScript installed to use this program anyway.

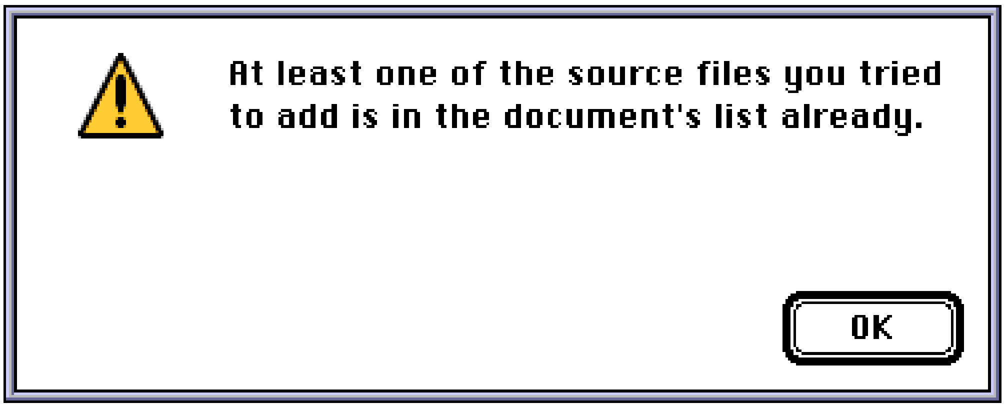
The "Drag and Drop" method

DocAssembler supports the Drag and Drop extension, that comes standard with Mac OS 7.5 and is available as an option with Mac OS 7.1.

If your system has the Drag and Drop extension installed, you will be able to collect the source files by selecting them in the Finder and then dragging them directly in a DocAssembler document window.
This is a much faster way to collect files compared to the "Open dialog" method, also because you are not limited to one file at the time. If you want, you can also drag and drop files between DocAssembler document windows.

If DocAssembler cannot accept one or more files, its action will be different depending on its current state: if it is the current active application, it will directly show an error message; if in the background, you will hear a "beep" and the application menu icon will flash until you will get DocAssembler in the foreground, when it finally will show the error message. If you are asking why all this complication, you should have seen it freeze my Macintosh trying to show an alert in the middle of a drag operation!

Error message when trying to add the same file twice


At least one of the source files you tried to add is in the document's list already.

The "Apple Script" method

Chances are that if you are a real power user, you will conclude that this is the most powerful method. Since DocAssembler supports the required, the core and the miscellaneous Apple Events suites and goes beyond, defining specific “Add”, “Produce”, “Set Preference” and “Set Output” events, you can create a document, fill it with source files and produce the output document in the format and with the attributes you prefer entirely using AppleScript.

See the “sample script” file in this package for an example on how to do this and turn to the "Apple Events reference" section to learn all about the events supported by DocAssembler.

How to structure the document

Hierarchical level of the source files

Each source file you collect in a DocAssembler document is assigned a "hierarchical" level between four different choices (0, -1, -2, -3). These levels could be identified, from top to bottom, with "section", "sub-section", "chapter", and "paragraph". You can use this option to accurately structure the output document.

When DocAssembler builds its list, it refers to the order and the level of each source file to assign it an indexing string, such as "1.1" or "2.3.1.4". You can then choose to include these strings in the output document. The appealing side of this feature is that you can apply repeated changes to both the order and the level of the source files while DocAssembler keeps mantaining the correctness of the indexing scheme.

When you use the "Add File..." command from the "Document" menu or the Drag and Drop method to collect the source files, their level is initialized to the "0" (top - section) level, while when you use an AppleScript to build your document, you can directly specify the level of each file.

To change the hierarchical level of a source file, first select it using the mouse or the keyboard, then choose the "Increase Level" or the "Reduce Level" commands from the "Document" menu. (These commands have button equivalents in the toolbar). Only one source file can be selected at once, so you cannot apply changes to a group of files simultaneously.

When one or both these commands aren't available, this means that the source file's level has reached the top or the bottom level allowed by the document context. For example, you can never change the level of the first file in the list, because it has to be at the topmost level; similarly, you can't assign a level which is more than one level deeper than the preceeding one.

How to change the source files order

Notice: this feature is available only if you have the Drag and Drop extension installed in your system (if you work with Mac OS 7.5, you have it).

To change the position of a file in the list, first select it with the mouse or using the arrow keys, then click again upon it and, without releasing the mouse button, drag it to the new position. Only one source file can be selected at once, so you cannot move a group of files simultaneously.

During this operation, a thin gray rectangle follows the mouse pointer, showing that you have grabbed something, while a thicker one shows the position the file would have if you released the mouse button.

If you don’t have the Drag and Drop extension, you can use an AppleScript to build the file list, changing the script until you obtain the desired results.

If you don’t have the AppleScript extension, your only chance is to carefully append each file in the right order using the “Add File...” command from the “Document” menu.

Headers and footers

According to the nature of this application, informations related to headers and footers in the source documents are not taken into account. Instead you should use the preferences dialogs to set them up in the output document.

How to access the source files

Editing files

DocAssembler doesn't allow you to directly edit the source files, but when you double-click on a source file in the list, the file's creator application, if available, is automatically launched and requested to open the file.

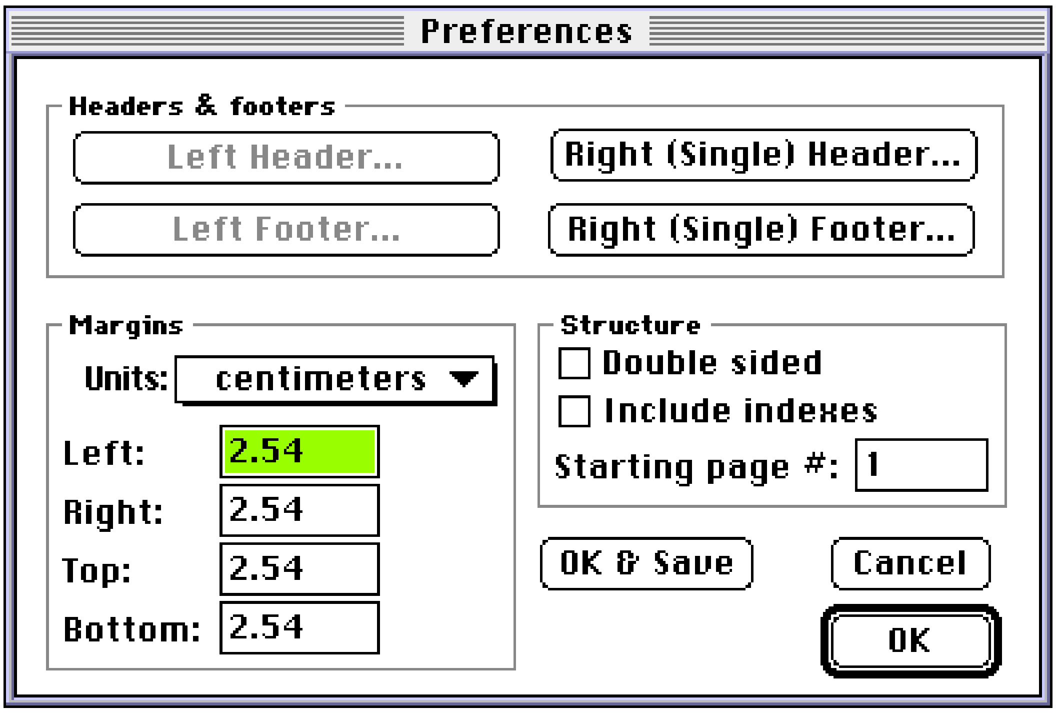
Tracking files

DocAssembler relies on the Alias Manager to keep track of the source files. This means that unless you do something really bad to those files, like erasing them, it will be able to track them even if you moved or renamed them.

Setting preferences

Using the "Preferences..." command from the "Document" menu, which also has a button equivalent in the toolbar, you can access a dialog to control several details about the output document.

The "Preferences" dialog


Preferences

Headers & footers

Left Header...

Right (Single) Header...

Left Footer...

Right (Single) Footer...

Margins

Units: centimeters

Left: 2.54

Right: 2.54

Top: 2.54

Bottom: 2.54

Structure

☐ Double sided

☐ Include indexes

Starting page #: 1

OK & Save

Cancel

OK

Besides the normal "OK" and "Cancel" buttons, there is a "OK & Save" button that, if pressed, will record the current settings in a preferences file in the preferences folder of your system folder. This file will then be used each time you create a new document to initialize its preferences. To revert to the default initialization, just delete the preferences file.

The main preferences dialog is divided in the following sections:

Margins

Here you can assign the left, right, top and bottom margin. Their values can be specified in centimeters, millimeters, inches or points, according to the status of the related popup menu.

Structure

If you check the "Include indexes" option, the strings showed at the left of each source file in the document window will be included in the output document immediately before the beginning of the source text and with the current text attributes. Thus, if your source files starts with the section title, the result will be that title precedeed by the index number in the same font, size, style and color. The default setting is checked.

Check the “Double sided” option if you want a document with facing pages. The default setting is checked. If you uncheck this option, please remember to refer to the “Right (single) header” and the “Right (single) footer” sections for header and footer settings.

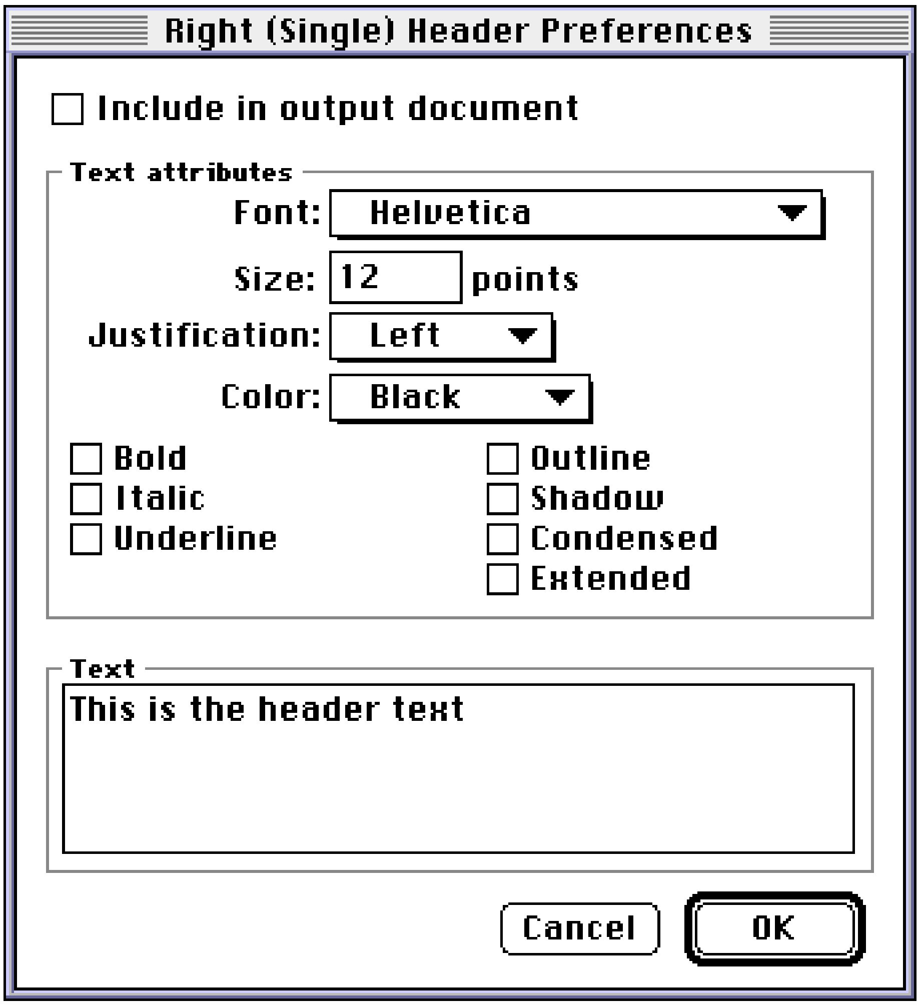
Use the “Starting page #” field to assign the number from which to start counting pages in the document. The default setting is 1.

Left header, Right (single) header, Left footer, Right (single) footer

These sections are perfectly analogous. Each one has a check box and a button. Check the “Include” option if you want to include the related header or footer in your document. The default setting is unchecked.

Use the “Edit...” button to activate a dialog to access the related header or footer characteristics.

The “Header / Footer preferences” dialog


Right (Single) Header Preferences

☐ Include in output document

Text attributes

Font: Helvetica

Size: 12 points

Justification: Left

Color: Black

☐ Bold
☐ Italic
☐ Underline
☐ Outline
☐ Shadow
☐ Condensed
☐ Extended

Text

This is the header text

This dialog has the following items:

- a “Font” popup menu that shows those available in your system;
- a “Size” field for the text height in points;
- a “Justification” popup menu that allows you to choose between left, center, right and full;
- a “Color” popup menu that lets you choose between the first eight standard colors defined by the XTND System;
- several check boxes to control the style of the text;
- a field to input the header's or footer's text.

Known bugs and problems

XTND limitations (nice pun, huh?)

Although the Claris XTND System is based upon a real powerful concept and deserves to be further developed, its implementation suffers from several shortcomings.

Obviously, if one formatting feature doesn't have its counterpart into the XTND architecture, it cannot be translated properly. This architecture is beginning to show its age: it knows nothing about features that have become almost standard today, like tables or borders; color management is crude at best, since all is mapped to the 8 standard XTND colors (white, black, red, green, blue, cyan, magenta and yellow), so you have to say goodbye to that delicate nuance that you liked so much...

Other problems occur due to the poor quality of the translators. I did several, I repeat SEVERAL, tests to discover that none of the translator available to me was capable to preserve <u>all</u> the characteristics of the original document.

Finally, although I was using the same set of translators, with some translator combinations I have obtained damaged files after an import-export cycle, where ClarisWorks has never shown such a behaviour. While is likely that the Claris development team is better than me at programming, I suspect they also have access to better documentation about XTND.

As you probably guessed, all this problems can drastically reduce the quality of the documents produced by this application. If you have read this far without giving up, I believe the best strategy is to experiment with the formatting of the original documents until a satisfying result is produced from the import-export translator combination. By the way, don't assume that the same translator will behave the same way both during import and export!

All that said, I know that have probably contributed with some exquisite bugs of my own. If you think you have discovered one or more of them, please let me know, even if you are not going to register.

Footnotes

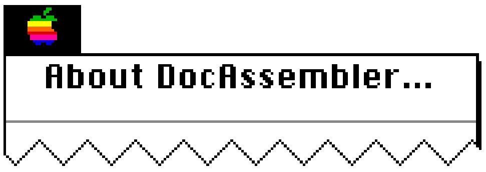
Sorry, but I decided to skip over footnotes, because I could not figure out an easy way to manage them using the XTND System with multiple files. I hope you would not consider this a major flaw.

AppleScript

Currently, errors encountered during the execution of a script are not managed properly, that is, the caller does not receive error information back. I hope to fix this soon.

Commands reference

The apple menu

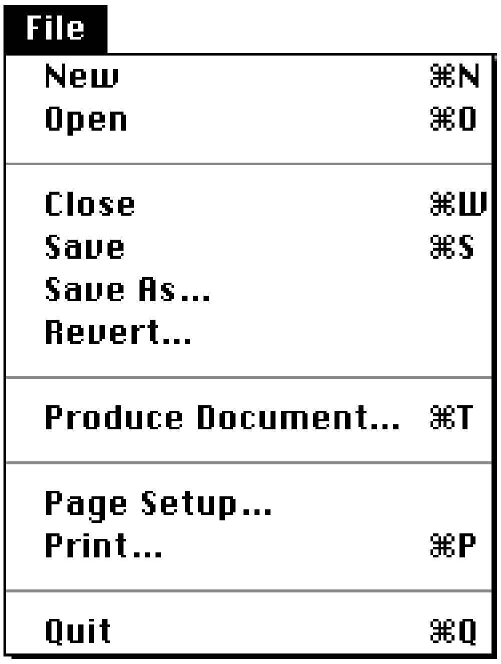


This menu contains a command to obtain informations about the DocAssembler application

About DocAssembler...

Shows a window that gives informations about the author, the version number, the name of the registered user... (If you use this program, please register!).

The File menu



This menu contains the standard file management commands, plus a custom one to generate the output file.

New

Creates a new empty DocAssembler document. This command has a button equivalent in the toolbar and is always available.

Open...

Opens an existing DocAssembler document. This command has a button equivalent in the toolbar and is always available.

Close

Closes the current active DocAssembler document. If the document's content aren't already saved, you have a chance to do that. This command is only available when there is at least one DocAssembler document currently opened.

Save

Saves the current active DocAssembler document. This command has a button equivalent in the toolbar and is only available when you made changes to the current DocAssembler document without saving them.

Save As...

Saves the current active DocAssembler document under a user choosen name. This command is only available when there is at least one DocAssembler document currently opened.

Revert...

Reloads the last saved version of the current active DocAssembler document, discarding any unsaved changes. This command is only available when you have made changes to the current DocAssembler document without saving them.

Produce Document...

Generates the output document, letting the user choose its format between the available XTND translators. This command is available when there is at least one entry in the current document's source file list.

Page Setup...

Shows the standard "Page Setup" dialog relative to the current selected printer in the chooser.
This command is always available.

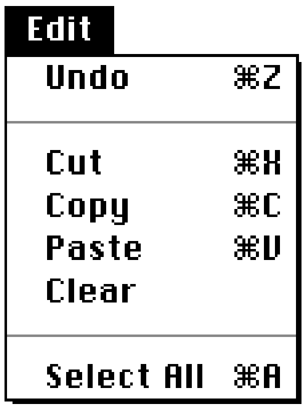
Print...

Shows the standard "Print" dialog relative to the current selected printer in the chooser. This command is only available when there is at least one DocAssembler document currently opened.

Quit

Quits DocAssembler. If you have any open document whose contents aren't already saved, you have a chance to do that. This command is always available.

The Edit menu



This menu contains the standard edit commands.

Undo

Undoes the last undoable operation. This can be a text editing operation in a dialog or the changes applied using the "Preferences" dialog. This command has a button equivalent in the toolbar, its availability and name change according to the context.

Cut

Clears the current selection, which can be made of text or a file record, and copies it to the clipboard. This command is only available when a selection exists.

Copy

Copies the current selection, which can be made of text or a file record, to the clipboard. This command is only available when a selection exists.

Paste

Copies the contents of the clipboard, which can be made of text or a file record, at the current insertion point. This command is only available when the clipboard's contents are appropriate to the context.

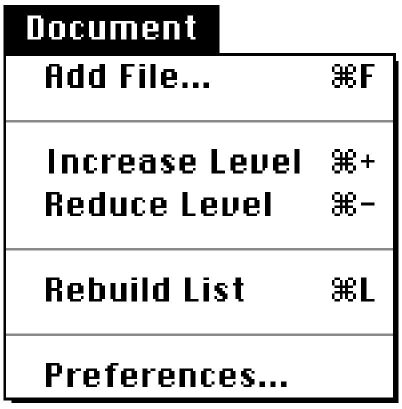
Clear

Clears the current selection, which can be made of text or a file record. This command is only available when a selection exists.

Select All

Selects all the text in an edit field. This command is only available when there is text to select.

The Document menu



Use this menu to append or remove source files, to change their hierarchical level and to set the options related to the output document.

Add File...

Activates the standard "Open" dialog to let the user choose a source file to append to the current DocAssembler document. If the choosen file is already in the list, a warning message is shown and the file is not appended. This command has a button equivalent in the toolbar and is only available when there is at least one DocAssembler document currently opened.

Increase Level

Increases the level of the currently selected source file in the list. This command has a button equivalent in the toolbar and is only available when a source file whose level is increasable is selected in a DocAssembler document.

Reduce Level

Reduces the level of the currently selected source file in the list. This command has a button equivalent in the toolbar and is only available when a source file whose level is reduceable is selected in a DocAssembler document.

Rebuild List

Checks for the existance of each file record in the list of the current DocAssembler document and verifies if there a proper XTND translator and the creator application are available.
This command has a button equivalent in the toolbar and is available when there is at least one file record in the currently opened DocAssembler document.

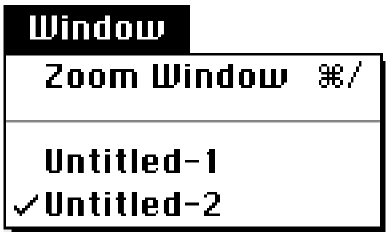
Preferences...

Activates a dialog to let the user choose among several options related to the output document. From this dialog you can:

- assign margins, choosing their units between centimeters, millimeters, inches or points;
- choose if the document will be single or double sided;
- choose if to include automatic indexing;
- assign the starting page number;
- choose if to include headers and / or footers and assign their text and attributes;

This command is only available when there is at least one DocAssembler document currently opened.

The Window menu



Use this menu to choose the active DocAssembler document or to change the zoom state of its window.

Zoom Window

Changes the zoom state of the current DocAssembler document window and is equivalent to click in the zoom box of the window. This command is only available when there is at least one DocAssembler document currently opened.

Apple Events reference

This section describes the Apple Event support in DocAssembler. It will explain each Apple Event which can be sent to the application. Description for the required suite is omitted, you can assume the standard behaviour.

The Standard Suite

Close

Closes an object.

Event Class: kCoreEventClass
Event ID: kAEClose

keyDirectObject:

Descriptor Type: cObjectSpecifier -- the objects to close
Required / Optional: required

Additional Parameters:

saving -- specifies whether or not changes should be saved before closing

Descriptor Keyword: keyAESaveOptions
Descriptor Type: typeEnumeration
Required / Optional: optional

in -- the file in which to save the object

Descriptor Keyword: keyAEFile
Descriptor Type: typeAlias
Required / Optional: optional

Reply Parameters:

«none»

AppleScript example:

```
tell application "DocAssembler 1.0" to close document "sample"
tell application "DocAssembler 1.0" to close window "sample"
tell application "DocAssembler 1.0" to close document saving yes in "sample"
```

Data Size

Returns the size in bytes of an object.

Event Class: kAECoreSuite
Event ID: kAEGetDataSize

keyDirectObject:

Descriptor Type:	cObjectSpecifier -- the object whose data size is to be returned
Required / Optional:	required

Reply Parameters:

Descriptor Type:	typeInteger -- the size of the object in bytes

AppleScript example:

```
tell application "DocAssembler 1.0" to get data size of document "sample"
```

Get

Gets the data for an object.

Event Class:	kAECoreSuite
Event ID:	kAEGetData

keyDirectObject:

Descriptor Type:	cObjectSpecifier -- the object whose data is to be returned
Required / Optional:	required

Reply Parameters:

Descriptor Type:	typeWildCard -- the data from the object

AppleScript example:

```
tell application "DocAssembler 1.0" to get document "sample"
```

Make

Makes a new element.

Event Class:	kAECoreSuite
Event ID:	kAECreateElement

keyDirectObject:

Descriptor Type:	typeNull
Required / Optional:	optional

Additional Parameters:

new -- the class of the new element
Descriptor Keyword: keyAEObjectClass
Descriptor Type: typeType
Required / Optional: required

at -- the location at which to insert the element
Descriptor Keyword: keyAEInsertHere
Descriptor Type: typeInsertionLoc
Required / Optional: optional

with data -- the initial data for the element
Descriptor Keyword: keyAEData
Descriptor Type: typeWildCard
Required / Optional: optional

with properties -- the initial values for the properties of the element
Descriptor Keyword: keyAEPropData
Descriptor Type: typeAERecord
Required / Optional: optional

Reply Parameters:

«none»

AppleScript example:

```
tell application "DocAssembler 1.0" to make new document "sample"
tell application "DocAssembler 1.0" to make new window "sample" at {50, 50}
```

Open

Opens the specified document.

Event Class:	kCoreEventClass
Event ID:	kAEOpenDocument

keyDirectObject:

Descriptor Type:	typeAEList -- can be a list of alias records or an object specifier
Required / Optional:	required.

Reply Parameters:

«none»

AppleScript example:

```
tell application "DocAssembler 1.0" to open document "sample"
tell application "DocAssembler 1.0" to open window "sample"
tell application "DocAssembler 1.0" to open document {"sample.1", "sample.2"}
```

Print

Prints the specified document.

Event Class:	kCoreEventClass
Event ID:	kAEPrintDocument

keyDirectObject:

Descriptor Type:	typeAEList -- can be a list of alias records or an object specifier
Required / Optional:	required.

Reply Parameters:

«none»

AppleScript example:

```
tell application "DocAssembler 1.0" to print document "sample"
tell application "DocAssembler 1.0" to print window "sample"
tell application "DocAssembler 1.0" to print document {"sample.1", "sample.2"}
```

Save

Saves a document.

Event Class:	kAECoreSuite
Event ID:	kAESave

keyDirectObject:

Descriptor Type:	cObjectSpecifier -- the objects to save
Required / Optional:	required

Additional Parameters:

in -- the file in which to save the object(s)
Descriptor Keyword: keyAEFile
Descriptor Type: typeAlias
Required / Optional: required

as -- the file type of the document in which to save the data
Descriptor Keyword: keyAEFileType
Descriptor Type: typeType
Required / Optional: optional

Reply Parameters:

«none»

AppleScript example:

```
tell application "DocAssembler 1.0" to save document "sample"
tell application "DocAssembler 1.0" to save document in "sample.2"
```

Set

Sets an object’s data.

Event Class:	kAECoreSuite
Event ID:	kAESetData

keyDirectObject:

Descriptor Type:	cObjectSpecifier -- the object to change
Required / Optional:	required

Additional Parameters:

to -- the new value
Descriptor Keyword: keyAEData
Descriptor Type: typeWildCard

Required / Optional: required

Reply Parameters:

«none»

AppleScript example:

```
tell application "DocAssembler 1.0" to set document title to "sample"
```

The "Odds" Suite

Select

Selects the specified object.

Event Class:	'Odds'
Event ID:	kAESelect

keyDirectObject:

Descriptor Type:	cObjectSpecifier -- the object to select
Required / Optional:	required.

Reply Parameters:

«none»

AppleScript example:

```
tell application "DocAssembler 1.0" to select document "sample"
tell application "DocAssembler 1.0" to select window "sample"
```

The Miscellaneous Standards Suite

Revert

Reverts a document to the most recently saved version.

Event Class:	kAEMiscStandards
Event ID:	kAERevert

keyDirectObject:

Descriptor Type:	cObjectSpecifier -- the object to revert
Required / Optional:	required

Reply Parameters:

«none»

AppleScript example:

```
tell application "DocAssembler 1.0" to revert document "sample"
```

The DocAssembler Suite

Add

Adds a source file to a DocAssembler document.

Event Class: kAE_DA_Suite
Event ID: kAE_DA_AppendFile

keyDirectObject:

Descriptor Type: cObjectSpecifier -- the document to add the file to
Required / Optional: required

Additional Parameters:

the -- specifies the file to add

Descriptor Keyword: keyAEFile
Descriptor Type: typeAlias
Required / Optional: required

with level -- the hierarchical level for the file

Descriptor Keyword: keyAELevel
Descriptor Type: typeShortInteger
Required / Optional: required

Reply Parameters:

«none»

AppleScript example:

```
tell application "DocAssembler 1.0"
        tell document "sample" to add the file "source" with level -1
end tell
```

Produce

Produces output from a DocAssembler document.

Event Class: kAE_DA_Suite
Event ID: kAE_DA_ProduceFile

keyDirectObject:

Descriptor Type: cObjectSpecifier -- the document to produce the file
Required / Optional: required

Additional Parameters:

the -- specifies the file to produce

Descriptor Keyword: keyAEFile
Descriptor Type: typeAlias

Required / Optional: required

using translator -- the translator to use

Descriptor Keyword: keyAE_DA_TransName
Descriptor Type: typeChar
Required / Optional: required

with format -- the requested format

Descriptor Keyword: keyAEFileType
Descriptor Type: typeType
Required / Optional: required

Reply Parameters:

«none»

AppleScript example:

```
tell application "DocAssembler 1.0"
        tell document "sample" to produce the file "sample.out" using translator "RTF" with
format "TEXT"
end tell
```

Set Preference

Chooses options for a DocAssembler document.

Event Class: kAE_DA_Suite
Event ID: kAE_DA_SetPrefs

keyDirectObject:

Descriptor Type: cObjectSpecifier -- the document whose preferences are being set
Required / Optional: required

Additional Parameters:

units to -- choose margin units between cm, mm, inches and points

Descriptor Keyword: 'PR01'
Descriptor Type: typeEnumeration
Required / Optional: optional

left margin to -- the left margin in the preferred units

Descriptor Keyword: 'PR02'
Descriptor Type: typeSMFloat
Required / Optional: optional

right margin to -- the right margin in the preferred units

Descriptor Keyword: 'PR03'
Descriptor Type: typeSMFloat
Required / Optional: optional

top margin to -- the top margin in the preferred units

Descriptor Keyword: 'PR04'
Descriptor Type: typeSMFloat
Required / Optional: optional

bottom margin to -- the bottom margin in the preferred units

Descriptor Keyword: 'PR05'
Descriptor Type: typeSMFloat
Required / Optional: optional

double sided -- to have facing pages

Descriptor Keyword: 'PR06'
Descriptor Type: typeBoolean
Required / Optional: optional

include indexes -- to include auto-generated indexes

Descriptor Keyword: 'PR07'
Descriptor Type: typeBoolean
Required / Optional: optional

starting page to -- the starting page number

Descriptor Keyword: 'PR08'
Descriptor Type: typeShortInteger
Required / Optional: optional

Reply Parameters:

«none»

AppleScript example:

```
tell application "DocAssembler 1.0"
        tell document "sample" to set preference units to inches
        tell document "sample" to set preference left margin to 1.5
        tell document "sample" to set preference right margin to 1.5
        tell document "sample" to set preference top margin to 2.0
        tell document "sample" to set preference bottom margin to 2.0
        tell document "sample" to set preference with double sided
        tell document "sample" to set preference with include indexes
        tell document "sample" to set preference starting page to 3
end tell
```

Set Output

Sets the preferences for headers or footers.

Event Class: kAE_DA_Suite
Event ID: kAE_DA_SetOutputOption

keyDirectObject:

Descriptor Type: cObjectSpecifier -- the document whose headers or footers are being set

Required / Optional: required

Additional Parameters:

option -- option code (identifies header or footer)

Descriptor Keyword: 'OpID'
Descriptor Type: typeEnumeration
Required / Optional: required

include -- include this header or footer

Descriptor Keyword: keyAE_DA_HFInclusion
Descriptor Type: typeBoolean
Required / Optional: optional

font to -- the name of the font

Descriptor Keyword: keyAE_DA_HFFont
Descriptor Type: typeChar
Required / Optional: optional

size to -- the size of the text in points

Descriptor Keyword: keyAE_DA_HFSize
Descriptor Type: typeShortInteger
Required / Optional: optional

style to -- the text style attribute

Descriptor Keyword: keyAE_DA_HFStyle
Descriptor Type: typeEnumeration
Required / Optional: optional

justification to -- the text justification

Descriptor Keyword: keyAE_DA_HFJust
Descriptor Type: typeEnumeration
Required / Optional: optional

color to -- the text color

Descriptor Keyword: keyAE_DA_HFColor
Descriptor Type: typeEnumeration
Required / Optional: optional

text to -- the text

Descriptor Keyword: keyAE_DA_HFText
Descriptor Type: typeChar
Required / Optional: optional

Reply Parameters:

«none»

AppleScript example:

```
tell application "DocAssembler 1.0"
	tell document "sample" to set output option left header with include
	tell document "sample" to set output option left header font to "Courier"
	tell document "sample" to set output option left header size to 14
	tell document "sample" to set output option left header style to underline
	tell document "sample" to set output option left header justification to center
	tell document "sample" to set output option left header color to blue
	tell document "sample" to set output option left header text to "Here goes the text"
end tell
```

Constants and AppleScript enumeration terminology

```
kAE_DA_Suite             = 'FRob'

kAE_DA_AppendFile        = 'AddF'
kAE_DA_ProduceFile       = 'Prod'
kAE_DA_SetPrefs          = 'Pref'
kAE_DA_SetOutputOption   = 'POUT'

keyAE_DA_TransName       = 'Tran'
keyAE_DA_OptionID        = 'OpID'

keyAE_DA_Units           = 'PR01'
keyAE_DA_LeftMargin      = 'PR02'
keyAE_DA_RightMargin     = 'PR03'
keyAE_DA_TopMargin       = 'PR04'
keyAE_DA_BottomMargin    = 'PR05'
keyAE_DA_DoubleSided     = 'PR06'
keyAE_DA_IncludeIndexes  = 'PR07'
keyAE_DA_StartingPage    = 'PR08'

keyAE_DA_HFInclusion     = 'HF01'
keyAE_DA_HFFont          = 'HF02'
keyAE_DA_HFSize          = 'HF03'
keyAE_DA_HFStyle         = 'HF04'
keyAE_DA_HFJust          = 'HF05'
keyAE_DA_HFColor         = 'HF06'
keyAE_DA_HFText          = 'HF07'

enumCentimeters          = 'uniC' • centimeters
enumMillimeters          = 'uniM' • millimeters
enumInches               = 'uniI' • inches
enumPoints               = 'uniP' • points

enumLeftHeader           = 'LeHe' • left header
enumRightHeader          = 'RiHe' • right header
enumLeftFooter           = 'LeFo' • left footer
enumRightFooter          = 'RiFo' • right footer

enumStylePlain           = 'plan' • plain
enumStyleBold            = 'bold' • bold
enumStyleItalic          = 'ital' • italic
enumStyleUnderline       = 'undl' • underline
enumStyleOutline         = 'outl' • outline
enumStyleShadow          = 'shad' • shadow
enumStyleCondense        = 'cond' • condense
enumStyleExtend          = 'exte' • extend
```

```
enumJustLeft            = 'jusL' • left
enumJustCenter          = 'jusC' • center
enumJustRight           = 'jusR' • right
enumJustFull            = 'jusF' • full

enumColorWhite          = 'colW' • white
enumColorBlack          = 'colN' • black
enumColorRed            = 'colR' • red
enumColorGreen          = 'colG' • green
enumColorBlue           = 'colB' • blue
enumColorCyan           = 'colC' • cyan
enumColorMagenta        = 'colM' • magenta
enumColorYellow         = 'colY' • yellow
```

Version history

1.0 (March 1996)

First public release.