Chapter1
Overview

Beginning with version 2.0, Arrange provides a powerful open architecture for plug-in modules, which can add new features or modify existing features of the application.

A plug-in module is a file that contains executable code. Plug-ins add functionality to Arrange. For example, plug-ins can define new menu commands, override the behavior of existing commands, add import or export support for new data formats, or customize the formatting of text fields.

1.1 About this document

This document is written for developers who will develop plug-in modules for Arrange. It contains descriptions of the following:

- Arrange data model
- Arrange API
- An Arrange plug-in

This document also describes how to create and debug an Arrange plug-in.

1.1.1 Who should read this document?

This document contains detailed technical information. It is aimed at a sophisticated audience. Typically, if you're going to write a plug-in, you should have the following background:

- can program in C or C++
- can use MPW™, MetroWorks Code Warrior™, or Symantek's C++/Think C™ development tools
- have developed software for the Macintosh
- have used Arrange or are familiar with the Arrange paradigm

1.1.2 How to use this document

This section describes each of the chapters in this document and contains some suggestions for how you should use this book as you create a plug-in. You should know how to use Arrange before you develop a plug-in. See the Arrange User Guide and the 2.0 Addendum to the User Guide for more information on the program.

- Chapter 2, “The Arrange data model,”describes the internal architecture of an Arrange document file. This chapter describes the data structures that your plug-in will manipulate through the API.
- Chapter 3, “The Arrange API,” describes the over 100 individual API callbacks that Arrange makes available to plug-in modules.
- Chapter 4, “What is a plug-in?,” describes how plug-in files are structured and the environment in which plug-ins are executed. It also describes the generic plug-in file and lists things that you should know about working with the Arrange user interface specification.
- Chapter 5 “Writing a plug-in,” describes the process to follow when you create a plug-in. It also describes how to use development tools when creating a plug-in and describes tips and techniques that can help you develop plug-ins.
- Chapter 6, “Debugging a plug-in,” describes how to debug your plug-in.

1.1.3 Terminology

This section contains a description of terminology that’s used throughout this document:

- A document is a single Arrange file, stored in native or TAIL format, that contains notes, field definitions, topics, and so forth.
- A folder is an object which contains a list of topics. (Internally, each document has a “root folder” which contains all the folders in the contents list. This folder contains a list of folders, not a list of topics.)
- A topic is an object which contains a list of notes.
- A view is a set of viewing options, such as a filter rule and a sorting rule, that is attached to a topic (or folder). A view controls how a topic’s notes are displayed.
- A document window is a window that shows information in a document. The catalogs (Note Catalog, etc.) are considered to be document windows (in addition to those windows that the user would think of as document windows).
- A note is an entity stored in a document. Each note has one or more fields and a note type.
- A type definition defines a particular note type, specifying the type’s icon, a minimal set of fields which all notes of that type must have, and so forth.
- A field definition defines a particular field and specifies its name, data type, and so forth.
- A field instance is the appearance of a field in a specific note. The field

definition contains information that applies to all of the field's instances.

- A visible field is a field instance that appears in a note when the note is opened. From the user's point of view, all field instances are visible fields, since these are the only field instances the user can see.
- An invisible field (also called a hidden field), is a field instance that does not appear visibly in a note. Examples are the system fields (Date Created, Date Modified, Creator Name, and Last Editor Name), which are are always present in all notes, whether explicitly added by the user or not. If the user adds them explicitly to a particular note (or its note type), then they are visible fields; otherwise they are hidden fields.

  Visibility is a property of a field instance, not the field definition. The Date Created field, for example, can be visible in one note and invisible in another.

1.2 Other sources of information.

The Arrange Plugin Module Developer's Kit includes several other sources of information for a plug-in module developer:

- The C interface (.h) files declare the functions and data types described in this file.
- Commented source code for several working modules, including a "generic module" which can be used as a template for your own modules.
- The TAIL Specification provides information on the TAIL language, used by Arrange to exchange data with the Grabber system extension and other external entities.