About the Script Manager and System 7.0

The **Script Manager** provides

- standard routines for the manipulation of ordinary text
- a means to make your application work with many writing systems
- access to and coordination with the set of routines known as the **International Utilities Package** to provide localizable date, time, and number conversion

New **Script Manager** features available with system software version 7.0 allow you to

- retrieve font and style information in each script's local variables
- determine if a double-byte script system is installed
- obtain a pointer to the current 'KCHR' resource
- determine the current region code
- obtain improved information on word boundaries for word selection and line breaking (using information that individual script systems supply with appropriate tables in the 'itl2' resource)
- perform more sophisticated and faster word selection and word wrap
- truncate text in a way that improves its adaptation to different scripts and languages
- substitute text in a way that improves its adaptation to different scripts and languages
- perform uppercase and lowercase conversion more easily
- strip diacritical marks
- handle fully justified text with intercharacter spacing and multiple style runs on a line, using special scaling if desired
- create simple script systems that use the Roman Script System
- use the Keyboard menu to select keyboard layouts
- install and remove multiple scripts, keyboards, and fonts
- use new KeyScript verbs to select the next available keyboard within a script, to restrict the available keyboards temporarily

The features that let you set and retrieve font and style information, determine if a double-byte script system is installed, determine the current region code, and obtain a pointer to the current 'KCHR' resource are described in **Checking and Modifying Global and Local Variables**.

The following sections supply some background on how the **Script Manager**

- uses local and global variables to set up an environment in which users can install multiple script systems
- allows your application to organize text into runs to accommodate more than one script on a line and in a document
- parses text into tokens so your application can recognize meaningful symbols without making script-specific assumptions
- lets you localize dates, locations, and times
- allows your applications to localize the display of formatted numbers

Local and Global Variables

The **Script Manager** maintains a number of global variables that your application can read with the **GetEnvirons** function. These variables can be set by the corresponding **SetEnvirons** function. In addition, each script system maintains variables of its own, called *local variables*. You can read and set the local variables using **GetScript** and **SetScript**.

The **Script Manager** uses the global variables to set up and maintain the environment. For example, the global variable smDoubleByte, new with system software version 7.0, indicates if a double-byte script system is installed. Some local variables provide information on how scripts work; others control how they operate. For example, the local variable scriptMonoFondSize, new with system software version 7.0, specifies the default mono-spaced font and its size.

See the descriptions of the **GetEnvirons** and **SetEnvirons** functions for detailed discussions of variables available prior to system software version 7.0.

Style Runs and Higher-Level Text Organization

All **Script Manager** text-handling routines are based on the concept of **runs** (that is, consecutive text with the same attributes) including style runs, script runs, and direction runs. The **Script Manager** organizes text into a hierarchy beginning with style (or format) runs. A **style run** is a sequence of text all in the same font, size, style, color, and script. A **script run** is a sequence of text all in the same script. A **direction run** is a sequence of text with characters having the same direction.

Tokens

Programs that parse text (for example, compilers and assemblers) usually assign sequences of characters to abstract categories called **tokens**-such as variable names, meaningful symbols, and quoted literals. The **IntlTokenize** function allows your application to recognize tokens without making assumptions that depend on a particular script. For example, a single token for less than or equal to might have two representations in the U.S. system software: the two-character sequence <= or the single-byte character ≤. The latter is not available in the Japanese system software, which instead uses a 2-byte coding for the single character ≤. The **IntlTokenize** function handles these details so your application need not be aware of the differences. The tokenizer identifies the different elements in an arbitrary string of text by

using localized information from the **'itl4'** resource. (The **'itl4'** resource contains the localized code and resources for the tokenizer. See the subheading, **The 'itl4' Resource**, under **Using the International and Keyboard Resources** for details.)

Certain symbols in the standard Roman character set were not supported in earlier versions of the system software and had no corresponding token types. With system software version 6.0.4, the **Script Manager** added five new token types for some of these characters. With system software version 7.0, two new token types have been added: tokenEllipsis and tokenCenterDot. The tokenEllipsis type is used for the character that indicates truncation. The **TruncText** and **TruncString** functions obtain the corresponding default canonical character from the untoken table in the **'itl4'** resource. The tokenCenterDot type is used for the various forms of the centered dot, such as the one used by AppleShare for echoing passwords. The corresponding default or canonical character can be obtained from the untoken table in the **'itl4'** resource.

Date Conversion

The Macintosh extended date routines can handle a range of roughly 35,000 years. If your application needs a large range of dates, you can use system routines rather than produce your own, which may not be compatible worldwide. Date and time conversion may depend upon geographic information. For details, see **Date and Time Utilities**.

Geographic Information

You can also access the stored location (latitude and longitude) and time zone of the Macintosh from parameter RAM. The Map control panel gives users the ability to change and reference these values. For details, see **Worldwide Control Panels and Desk Accessory** under the section, **Related WorldWide Components**, and **Reading and Storing Locations** under **Overview of the Script Manager Routines**.

Number Conversion

The **Script Manager** number routines supplement the Standard Apple Numerics Environment (SANE) and allow applications to display formatted numbers and to read both formatted and simple numbers. The formatting strings allow display and entry of numbers and editing of format strings, even though the numbers and the format strings may have been entered using different localized system software. For brief descriptions of the number routines, see **Number Utilities** under the section entitled, **Overview of the Script Manager Routines**. For a thorough treatment of number conversion, see *Macintosh Worldwide Development: Guide to System Software*. .