

CalendarSet™

User Manual



CalendarSet

User Manual

Version 3.6

e-Node
30 rue de la République
33150 Cenon
France

www.e-node.net

Copyright and Trademarks

All trade names referenced in this document are the trademark or registered trademark of their respective holders.

CalendarSet is copyright Beckware LLC and exclusively published worldwide by e-Node.

4th Dimension, 4D Compiler, 4D, 4D Server, 4D Client, and 4D Insider are trademarks of 4D SA.

Windows, Excel and Vista are trademarks of Microsoft Corporation.

Macintosh, MacOS and MacOS X are trademarks of Apple, Inc.

Compatibility Information

CalendarSet is fully compatible with 4D/4D Server 2004 or greater (including 4D v11 SQL). It is compatible with MacOS and Windows clients.

Technical Support

Technical support for CalendarSet will be provided electronically via e-mail or our online support reporting system. You are encouraged to use the online web reporting form as it will be correctly routed to the appropriate support personnel.

www.e-node.net

Table of Contents

SoftwareCopyright and Trademarks -	
Compatibility information - Technical support.....	iv
About CalendarSet.....	8
Installing CalendarSet.....	9
Installation: Plug-In bundle (MacOS & Windows).....	9
Registration - License types.....	10
The CalendarSet User Interface.....	11
Word Wrap	11
Selecting Days	11
Event and Banner Selection.....	12
Event Scrolling in a Day's Cell	12
Banner Resizing.....	13
Event and Banner Dragging.....	13
Dragging Events and Banners within a CalendarSet Object ..	13
Dragging Events out of a CalendarSet Object	14
Dragging Events into a CalendarSet Object	14
Intelligent Banner and Icon Drawing	14
Date and Time Selection Popups.....	15
Date Popup.....	15
Time Popup	15
Color Selection Popup	16
Icon Selection Popup	17
Developing with CalendarSet.....	19
Creating a CalendarSet object on a Layout	19
To configure a variable object as a CalendarSet object.....	19
Using the CalendarSet Commands.....	21
Command Descriptions and Syntax.....	21
Registering CalendarSet.....	22
Configuration Commands	23
Configuring CalendarSet Using the Advanced Properties Dialog	
23	
To Display the Advanced Properties Dialog.....	23
Setting the Data to Display	24
Setting Options	25
Formatting Options	26
Dragging	27
Preview	27
Configuring CalendarSet Using CalendarSet Commands	28
Setting Events, Banners and Icons.....	28
Setting Icons	28
Setting Color, Font, Size, and Style Attributes.....	29
Displaying the Month Name.....	29
Displaying Out of Range Days.....	29
Extending the CalendarSet Frame	29

Specifying the First Day of the Week.....	29
Calendar Colors.....	29
Day Selection.....	30
Event Selection.....	30
Banner Selection and Resizing.....	30
Word Wrap.....	31
Event Markers.....	31
Hiding Days.....	31
Resizable Calendar Areas on a Form.....	31
To Create a Resizable Calendar Object Using 4D v3.....	32
Commands.....	32
User Action Commands.....	52
Responding to User Actions on a CalendarSet Object.....	52
CalendarSet Dragging Commands.....	60
Background.....	60
Technical Details of the Dragging Implementation.....	60
What are access “codes”?.....	61
After the drag.....	62
CalendarSet on Multi-page Layouts.....	63
CalendarSet DataType.....	63
Popup Commands.....	74
Using the Date and Time Selection Popups.....	74
Using the Color Selection Popup.....	74
Using the Icon Selection Popup.....	75
Commands.....	75
CalendarSet Utility Commands.....	84
International Utilities.....	84
Array Intersection.....	84
CalendarSet on Multi-page Layouts.....	84
CalendarSet in an plug-in Window.....	90
Using CalendarSet in an plug-in Window.....	90
CalendarSet and Printing.....	92
CalendarSet Examples.....	93
Example 1 — Create a Simple Calendar.....	93
Example 2 — Display Database Data in a Calendar.....	94
Example 3 — Displaying Banners on a Calendar.....	96
Example 4 — Responding to User Actions.....	97
Example 5 — Dragging Events within a CalendarSet Object.....	98
Example 6 — Dragging Events to another CalendarSet Object on the Same Layout.....	99
Example 8 — Dragging Events to a CalendarSet Object on a Dif- ferent Layout.....	101
Example 9 — Dragging from AreaList Pro to CalendarSet ..	102
Troubleshooting.....	104
CalendarSet is not being updated properly.....	104
CalendarSet does not respond to single or double-clicks.....	104
CalendarSet™ Demo.....	105

About CalendarSet

CalendarSet is an easy-to-use tool for implementing calendars on 4th Dimension layouts. Because CalendarSet is an plug-in, it is very fast, and provides capabilities not available to the developer using native 4D commands and objects. Operation is extremely fast, and follows the Macintosh interface.

Data is passed to CalendarSet using 4D arrays, providing a simple method of displaying date information from your database. The arrays will typically be loaded using the 4D **SELECTION TO ARRAY** command, and displayed with only one or two CalendarSet commands.

Special tools are implemented for the developer who desires to customize the appearance and configuration of CalendarSet, allowing the customization to be implemented rapidly. Many options are available to control the behavior and appearance of CalendarSet, with a minimum of programming.

CalendarSet can also be displayed as an independent, resizable plug-in window.

CalendarSet includes a collection of plug-in areas and commands to implement popup menus for display of date, time, color, and icon data. You can use these capabilities to enhance your user interface both with the CalendarSet plug-in area as well as other tasks.

CalendarSet is fully compatible with 4D / 4D Server 2004 or greater (including 4D v11 SQL). It is compatible with MacOS and Windows clients.

Installing CalendarSet

This chapter outlines the steps necessary for installing CalendarSet into your existing applications.

CalendarSet must be installed (and de-installed) using the bundle installation method described herein.

Installation: Plug-In bundle (MacOS & Windows)

CalendarSet is provided as a plug-in bundle for 4D 2004, 4D v11 SQL or higher.

This single version will work with MacOS and Windows deployments (you don't need a separate MacOS and Windows versions).

- 1 — Locate the folder where CalendarSet has been installed on your computer.
- 2 — Locate the 4th Dimension structure where you wish to install the CalendarSet plug-in.
- 3 — If you don't already have a directory labeled "Plugins", create one now.
- 4 — Copy the following plug-in to your applications Plugins folder: CalendarSet.bundle.

Registration

CalendarSet requires a registration key to “unlock” the product making it a full working version. Call the **CS_Register** command (see CS_Register on page 32 for complete details) in the *On Startup* method.

Without the registration key, CalendarSet will operate in demonstration mode during 20 minutes.

Version 3.6 introduced a new license design. Previous licenses will not work with this release.

In order to activate CalendarSet 3.6 and above, you need to require a new license key from e-Node.

Upgrades from version 3.5 to version 3.6 are provided for free to all registered users (proof of purchase will be required if the previous license was not purchased from e-Node).

License types

Like all e-Node plug-ins, CalendarSet offers six different license types. There are no such things as MacOS vs Windows or Development vs Deployment:

- **Single user license.** This license allows development (interpreted mode) or deployment (interpreted or compiled mode) on 4D Standalone or Runtime. Since the registration key is linked to a specific 4D license, you need to provide the number returned by the 4D command **GET SERIAL INFORMATION** (first parameter). A new license will be supplied for free at any time if you change your 4D version and/or get a new 4D registration key, provided that your previous licenses match the current public version at the exchange time.
- **Small server.** This license allows development (interpreted mode) or deployment (interpreted or compiled mode) on 4D Server up to 10 users. The registration key is linked to your 4D Server license just as above.
- **Medium server.** This license allows development (interpreted mode) or deployment (interpreted or compiled mode) on 4D Server up with 11 to 20 users. The registration key is linked to your 4D Server license just as above.
- **Large server.** This license allows development (interpreted mode) or deployment (interpreted or compiled mode) on 4D Server over 20 users. The registration key is linked to your 4D Server license just as above.
- **Unlimited Single User.** This license allows development (interpreted mode) or deployment (interpreted or compiled mode) on as many 4D Standalone, Runtime or Engine copies that run your 4D application(s). This is a yearly license, which expires after the date when it is to be renewed. The expiration only affects interpreted mode. **Compiled applications using an obsolete license will never expire.**
- **Unlimited OEM.** This license allows development (interpreted mode) or deployment (interpreted or compiled mode) on as many 4D Server (of any number of users), 4D Standalone, Runtime or Engine copies that run your 4D application(s). This is a yearly license, which expires after the date when it is to be renewed. The expiration only affects interpreted mode. **Compiled applications using an obsolete license will never expire.**

A 4D database used to retrieve your 4D serial information is available from the following link:

<http://www.e-node.net/ftp/GetSerialInfo>

The CalendarSet User Interface

CalendarSet displays a calendar area on 4D layouts, such as shown below.



Figure 1

Word Wrap

CalendarSet will optionally word wrap event text to fit within a day's width, rather than truncating text which doesn't fit. Each event may be prefixed by a dash or a bullet to distinguish different events within a day.

No word wrapping will be performed for banners.

The **CS_SetEventOpts** command ([page 42](#)) is used to control the word wrap feature.

Selecting Days

Days on the calendar can be selected in the following ways:

- 1 Clicking on a day.
- 2 Shift-Click on a day. This will select all days between the day clicked on and any other selected days.
- 3 Command-Click on a day. This will toggle the selection of the day clicked on.

CS_Options ([page 41](#)) can set the selection mode to single day, multiple day, and discontinuous day selection capability. This command is also used to change the way that a day is highlighted.

Event and Banner Selection

A user can select events or banners. Currently, only a single event or banner can be selected. When selected, the item will be highlighted with the system highlight color set in the Color Control Panel. Days or events can be selected, but not both simultaneously.

Icons can not be selected.

CS_SetEventOpts ([page 42](#)) is used to control the selection mode. See [“Event Selection” on page 30](#) for a discussion of the different commands available.

Event Scrolling in a Day’s Cell

In some instances, the combined height of all the events within a given day may exceed the available space within the day’s cell due to the number of events or the applied font styles. In this case, the user may scroll the list of events in order to view and select events that are obscured from view. The scrolling will be accomplished by clicking up and down arrows within the day’s cell. The following picture illustrates where the scroll arrows will be drawn.

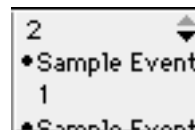


Figure 2

The arrows will be drawn regardless of the current day selection.

There is no developer interface to control vertical event scrolling, and it will be automatically invoked only in the cases described above.

The act of scrolling does not affect the current day or event

selection. The user can scroll a day other than the currently selected day.

The scrolled position is reset to the top only when the CalendarSet area is re-configured. This happens under the following scenarios:

- ◆ **AreaRefresh** ([page 88](#)) is called.
- ◆ The area is re-sized (plug-in window only).
- ◆ **CS_SetRange** ([page 34](#)) is called.

Banner Resizing

A banner can be resized by clicking on the end of the banner and dragging to a different day. During the resize, the cursor will change and the destination day will be highlighted. The end of a banner cannot be resized to a day prior to the start day and vice versa.

This feature is controlled with the **CS_SetBanrOpts** ([page 43](#)). You can detect that a banner has been resized using **CS_GetAction** ([page 53](#)), and determine information about the banner resize using **CS_GetResizBanr** ([page 57](#)).

CalendarSet v2 requires the Macintosh Drag and Drop extension to perform banner re-sizing. See [“Installation: Plug-In \(Macintosh\)” on page 9](#) and [“Banner Selection and Resizing” on page 30](#) for more information.

Event and Banner Dragging

CalendarSet lets the user drag events and banners within a CalendarSet object, as well as to other CalendarSet objects. The capability also allows dragging of events to AreaList Pro and other plug-in areas which support the 4D plug-in Dragging Interface Specification. See [“Technical Details of the Dragging Implementation” on page 60](#).

Dragging Events and Banners within a CalendarSet Object

When enabled, the user can select and drag events (and banners) within a CalendarSet object. An outline of the event will follow the pointer (cursor) location until the mouse is released. Only events (and banners) can be dragged. Days cannot be

dragged.

Dragging Events out of a CalendarSet Object.

When enabled, the user can select and drag events (and banners) out of a CalendarSet object. An outline of the event will follow the pointer (cursor) location until the mouse is released. Only events (and banners) can be dragged out of a CalendarSet object. Days cannot be dragged. The dragged items can be dragged to any open 4D window which has a drag-aware object configured to accept the drag.

Dragging Events into a CalendarSet Object.

When enabled, the user can drop items on both days and events. When over a day, the user will receive visual feedback via a “focus” rectangle that will be drawn within the target day’s cell (similar to the Finder). When over an event or banner, the user will see the event or banner highlight with the currently selected highlight color.

CalendarSet v2 requires the Macintosh Drag and Drop extension for event and banner dragging. [See “Installation: Plug-In \(Macintosh\)” on page 9.](#)

[See “CalendarSet Dragging Commands” on page 60](#) for a complete discussion of the dragging features and commands.

Intelligent Banner and Icon Drawing

Any events drawn on the same day/cell as a banner must be drawn differently in order to allow event selection and proper scrolling. Banners are drawn at the bottom of the day’s cell; however, space will be reserved between the day number and the banner area for regular events on a day in order to allow event selection and scrolling. The space will be equal to a single line of text drawn at 9 point Geneva. If one or more banners cannot fit in the space below, then all banners will be truncated to a height equivalent to 9 point Geneva. If this reduction is still insufficient to all the banners, then a scaling algorithm will be applied and each banner will be shown with a reduced vertical height with a minimum of 1 pixel of height (not counting the border).

Icons will be drawn before all other events to make it appear behind all other events. If other events or banners are shown on the same day as an icon, the event or banner will be drawn on

top of the icon.

Date and Time Selection Popups

CalendarSet includes two popups for user selection of date and time values. The popups appear on a layout with a triangle symbol

bol 

Date Popup

When the user clicks a date popup object, the following popup is displayed.



Sun, Apr 22, 2001							January	▲
S	M	T	W	T	F	S	February	1997
1	2	3	4	5	6	7	March	1998
8	9	10	11	12	13	14	April	1999
15	16	17	18	19	20	21	May	2000
22	23	24	25	26	27	28	June	2001
29	30						July	2002
							August	2003
							September	2004
							October	2005
							November	2006
							December	▼

Figure 3

There are three active areas on the popup, which allow selection of a year, month, and day.

[See “Using the Date and Time Selection Popups” on page 74](#) for a complete discussion of the date popup object and commands.

Time Popup

Clicking on a time popup object displays this popup:

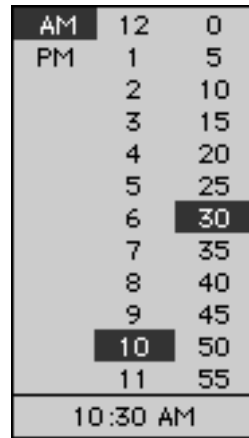


Figure 4

The three active areas on a time popup are for selection of hours, minutes, and AM/PM. The popup can be configured to display time in 24-hour format.

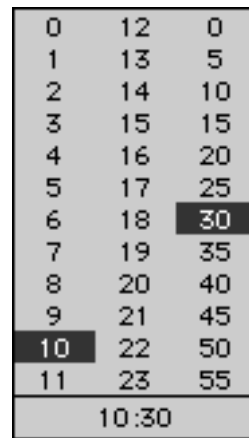


Figure 5

[See “Using the Date and Time Selection Popups” on page 74](#) for a complete discussion of the time popup object and commands.


Color Selection Popup

A color selection popup is provided as a method of selecting a color which can be assigned to an item or banner on a calendar. If the Mac is running in 256 color mode, the color selection popup displays the currently selected color, and when the user

clicks on the popup, a palette of eight standard colors and 4D's built-in 256 colors. The palette will display few colors if the Mac is running in a mode with fewer than 256 colors. When running in black and white mode, the popup displays the names of the eight standard colors, rather than a palette.



Figure 6

The palette also has a darkened rectangle to indicate the currently selected color .

[See “Using the Color Selection Popup” on page74](#) for a complete discussion of the color popup object and commands.

Icon Selection Popup

CalendarSet includes an icon selection popup object to provide a method for users to assign an icon to a day on a calendar, as shown below.



Figure 7

The popup displays the currently selected icon, and when the user clicks on the popup, a palette of icons is displayed. The icons are kept in the resource fork of the structure file, providing you or a sophisticated user with easy access for additions or modifications of icons.

[See “Using the Icon Selection Popup” on page 75](#) for a complete discussion of the icon popup object and commands.

Developing with CalendarSet

Creating a CalendarSet object on a Layout

Implementing CalendarSet in your 4D databases is very easy; in fact, displaying data in a CalendarSet area can be accomplished with only one plug-in command. The CalendarSet object is drawn on a 4D layout using the variable object tool. 4D opens the Definition dialog for the object, which is where the object is named and configured. The name will be used as a parameter for the CalendarSet commands.

Be careful to never have two CalendarSet objects with the same name on a 4D layout.

To configure a variable object as a CalendarSet object

- 1 Create a variable object on a layout.
4D displays the variable object configuration dialog.
- 2 Select the Plug-in Area type.
The popup below the Type popup will include the `_CS_Area` plug-in.
- 3 Choose the “`_CS_Area`” item from the popup below the Type popup.
4D will enter it as the Plug-in Routine name on the dialog, as shown below.
- 4 Name the variable.
This name will be used as the first parameter to many of the CalendarSet commands.
Note: This variable must be a process variable, not an interprocess variable (i.e., the name cannot begin with the “P” character) or local variable (i.e., the name cannot begin with the “\$” character).

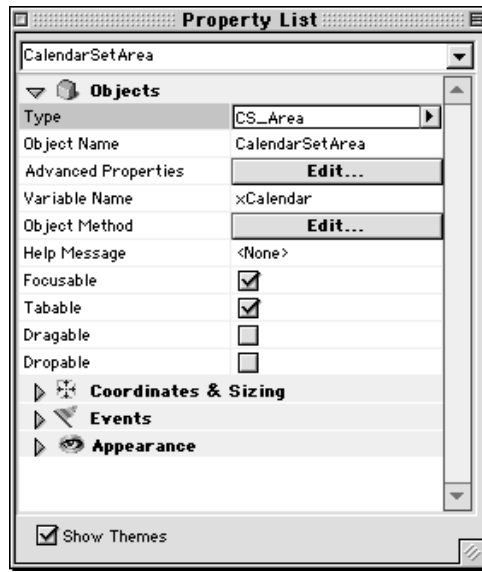


Figure 8

- Click the OK button.
The CalendarSet object is drawn in the layout editor.



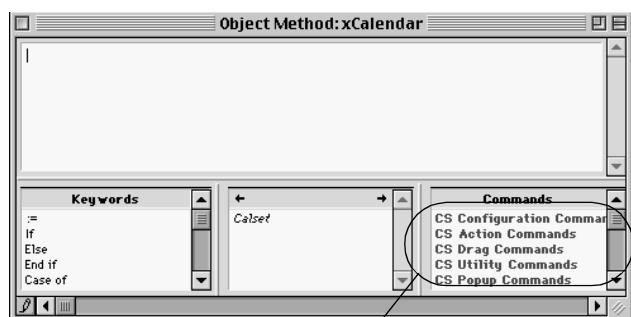
Figure 9

The first line is the name of the object, the remaining lines are the Copyright notice. Note the plug-in area shows the current month, to provide you with feedback about the sizing of the calendar area.

The display of the object name, Copyright notice, and current month is an indication that the object has been properly named and configured.

Using the CalendarSet Commands

The CalendarSet Commands are used in the same way that a 4D command is used. Parameters are separated by the semicolon character (“;”). You can access the CalendarSet commands in the Procedure editor list. Near the bottom of the list, below the area which contains the global procedures, there are seven CalendarSet command topics as shown below.



Click on a CalendarSet command topic to display a popup list of commands for that topic

Figure 10

Clicking on a topic presents a popup menu of the CalendarSet commands available. Simply select a command, and 4D will enter it for you at the current cursor position.

You can also type the command directly into the procedure.

Command Descriptions and Syntax

Each CalendarSet command has a syntax, or rules, that describe how to use the command in your 4D database. For each command, the name of the command is followed by the command’s parameters. The parameters are enclosed in parenthesis, and separated by semicolons. Following the command syntax description, an explanation of the command’s parameters is provided. For each parameter, the type of the parameter and a description is shown. Several examples are provided for each of the commands, showing examples of the syntax as well as how the various commands are used together.

The first parameter for each command is the name of the CalendarSet object on the layout. This parameter is a long integer, and is required to allow the commands to operate on the correct object.

Some parameters are used by CalendarSet to return a value. Do not

use local variables for these parameters, because 4D doesn't support returning values from an plug-in into a local variable. Use a process or interprocess variable.

Registering CalendarSet

CalendarSet must be registered with a serial number in order to function in normal mode. This number is included with your purchase of CalendarSet. If a valid serial number is not provided, then CalendarSet will function in demo mode. [See "CS_Register" on page 32](#) for complete details.

Configuration Commands

Configuring CalendarSet Using the Advanced Properties Dialog

CalendarSet v3 includes a point-and-click interface for configuring a CalendarSet object from within the Design environment. This dialog provides access to configure nearly every feature available via CalendarSet commands, and is very easy to use.

Note: the Advanced Properties Dialog is only available with 4th Dimension v6 or later. When using an earlier version of 4D, please refer to [“Configuring CalendarSet Using CalendarSet Commands” on page 28](#).

The Advanced Properties dialog lets you specify the names of the arrays to be displayed, almost all options including event/banner selection, color settings, and default styles and color for all CalendarSet text. There is a preview tab to instantly view the options that you've selected.

Once you click the OK button to complete the configuration, the settings will be saved by 4D within the plug-in area object on your layout. Whenever this layout is opened in the user/runtime environments, the settings made here will be applied to your CalendarSet object before the layout method or any object methods are executed. Essentially, you are replacing the default settings provided by CalendarSet with new values of your choosing. This functionality (dialog window, settings at runtime) is completely 4D v6-only.

You can use commands in combination with the Advanced Properties Dialog. In this case, CalendarSet first reads the settings specified in the dialog, then uses the settings specified by commands.

To Display the Advanced Properties Dialog

- 1 Double-click a CalendarSet object in the Form editor. 4D will display the Object Properties palette.

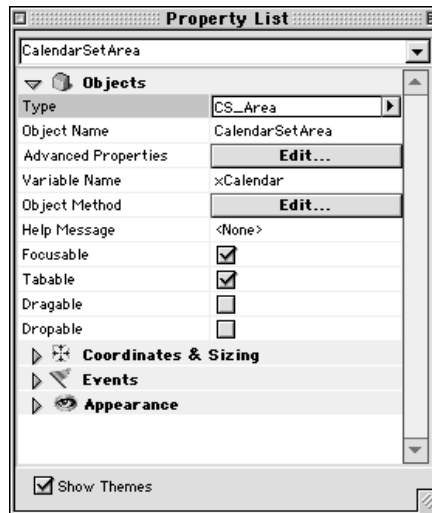


Figure 11

- 2 Click the Advanced Properties button.
The Advanced Properties Dialog will be displayed.

The dialog has several panes, accessed via the tabs at the top, which provide access to the various configuration options.

Note: in the sections following, hypertext “buttons” have been placed over areas of the dialog. You can click on any active object shown on that dialog to navigate to a section of the manual which explains the setting in detail.

Setting the Data to Display

Data is passed to CalendarSet via 4D arrays. You can tell CalendarSet the names of the arrays using the first pane on the Advanced Properties Dialog.

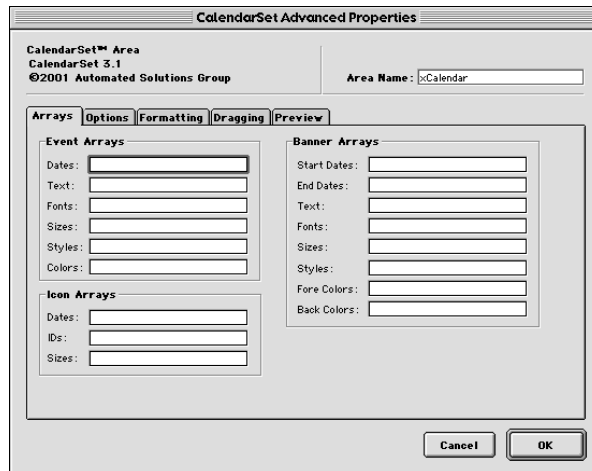


Figure 12

These arrays must be loaded from within your 4D methods. Typically, the arrays will be loaded using the 4D **SELECTION TO ARRAY** command in the On load event in the CalendarSet object's method. We recommend that you use the appropriate compiler declarations for all arrays you display, to ensure that you correct specify the data type of all arrays.

Refer to the 4th Dimension manuals for the details on defining arrays, and loading them with data.

Setting Options

The second pane on the Advanced Properties Dialog is used to set many of the options available for the display and behavior of the CalendarSet object. You can see the results of any settings you make by clicking on the Preview tab, to view a sample object.

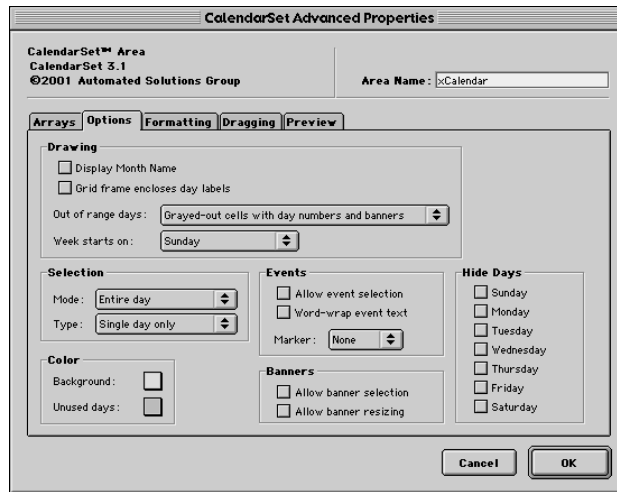


Figure 13

Formatting Options

CalendarSet lets you configure the font, size, style, and color used to display the various textual items on a CalendarSet object. Each of the types is accessed by the tab within this formatting pane. You can see the results of any settings you make by clicking on the Preview tab, to view a sample object.

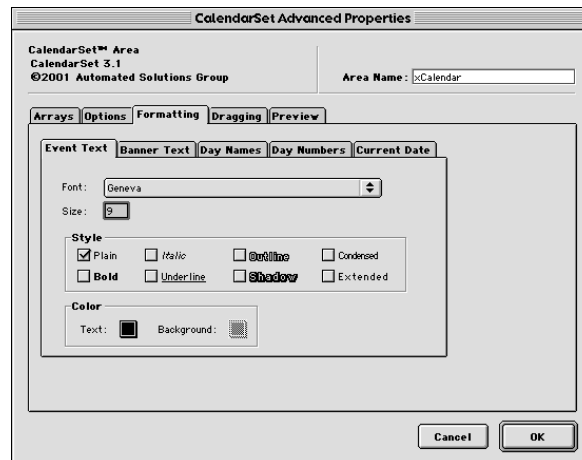


Figure 14

Dragging

CalendarSet supports drag-and-drop between CalendarSet objects, as well as with AreaList Pro objects. You can configure the dragging behavior using the Dragging pane of the Advanced Properties Dialog. *Please read the section [“Technical Details of the Dragging Implementation”](#) on page 60 for more information.*

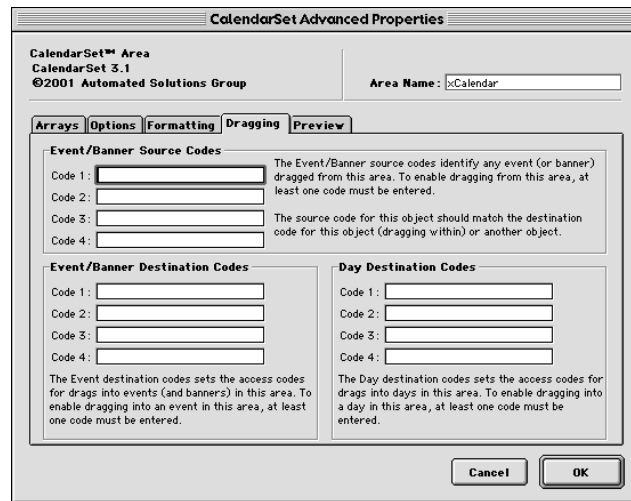


Figure 15

Preview

CalendarSet’s Advanced Properties Dialog provides a pane for previewing the current configuration settings. This is useful during the configuration process, as you can quickly see the results of any setting you select.

The Preview pane shows sample data using the settings you have specified in the Options, Formatting, and Dragging panes. The arrays you specify can’t be displayed, due to 4D’s process architecture, which keeps the Design environment in a special process.

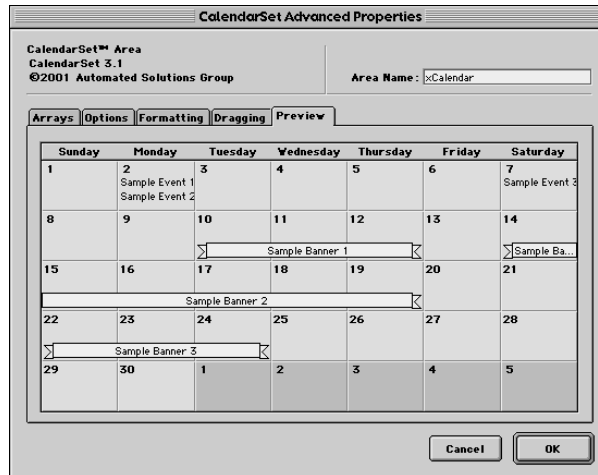


Figure 16

Configuring CalendarSet Using CalendarSet Commands

CalendarSet commands are used in the **Before** and **During** phase of the 4D execution cycle. A CalendarSet object is initialized in the **Before** phase as the layout is about to be displayed. Typically, this initialization will be contained in the script for the CalendarSet object. If you use a script with a CalendarSet object, and you want the script to execute in the Before phase, be sure to turn off the “Only if Modified” checkbox in the object’s definition dialog.

Setting Events, Banners and Icons

4D arrays are passed directly to CalendarSet for display, using **CS_SetArray** (page 35) for events, **CS_SetIconArray** (page 40) for icons, and **CS_SetBannerArray** (page 38) for banners. The arrays contain the data to be displayed on the days, as well as font, size, style, and color configuration. All arrays passed to a CalendarSet object must have the same number of elements.

Note: some arrays are passed as a string which is the array name, rather than just the array itself. In other words, the array name is enclosed in quotes. This approach is taken to allow CalendarSet to automatically handle certain tasks involved in redrawing a calendar area.

Setting Icons

CalendarSet allows display of an icon in a day (cell).

CS_SetIconArray ([page 40](#)) specifies the icons to display, along with the size of the icons.

Setting Color, Font, Size, and Style Attributes

The default foreground colors for each type of object on a CalendarSet area can be specified using **CS_FontDefaults** ([page 47](#)). The types of objects are day numbers, items, banners, day names, and current day number. Additionally, you can specify a unique color for a particular day number, item, or banner, using **CS_SetDayStyle** ([page 49](#)). These commands are also used to control font, size, and style.

Banners also have a background color attribute which can be set using the commands discussed above.

Displaying the Month Name

You can display an abbreviation of the month name on the first day of each month, using **CS_Options** ([page 41](#)).

Displaying Out of Range Days

Out of range days, such as the days before and after the month being displayed, can be displayed in several different ways, using **CS_Options** ([page 41](#)).

Extending the CalendarSet Frame

The border of the calendar may be extended to include the day headers (days of the week) using **CS_Options** ([page 41](#)).

Specifying the First Day of the Week

CalendarSet will default to displaying Sunday as the first day of each week. You can change this using **CS_Options** ([page 41](#)).

Calendar Colors

The colors of the entire calendar may be set using **CS_SetCalColor** ([page 44](#)). The unused days (those outside the date range specified for display using **CS_SetRange** ([page 34](#))) are drawn in the foreground color. The entire background of the calendar is shown in the background color.

Note: Icons colors may appear distorted when using background colors other than white.

Day Selection

You can control whether the user can click to select days, using ***CS_Options*** ([page 41](#)). Choices include the selection highlight mode, using *DayHighlightMode*, and the type of selection, using *DaySelectionType*. Highlight modes are no selection, day number only highlights, or entire day highlights. Selection types are no selection, or multiple contiguous or discontinuous days to be selected.

Event Selection

You can configure what type of selection to allow using ***CS_SetEventOpts*** ([page 42](#)). The parameter *AllowEventSelect* will consist of two values:

- 0 no event selection
- 1 single event selection

[See “Responding to User Actions on a CalendarSet Object” on page 52](#) for information about how to handle the selection of an event.

Banner Selection and Resizing

Banner selectibility can be set using ***CS_SetBanrOpts*** ([page 43](#)).

You can enable or disable banner resizing by setting *AllowBanner-Resize* using ***CS_SetBanrOpts*** ([page 43](#)).

When banner resizing is allowed, CalendarSet will automatically update the appropriate elements in the banner arrays that were passed using ***CS_SetBanrArray*** ([page 38](#)). [See “Responding to User Actions on a CalendarSet Object” on page 52](#) for information about how to handle the selection of a banner.

The user’s action of resizing a banner is communicated to you using ***CS_GetAction*** ([page 53](#)). This command should be called whenever the CalendarSet script is executed (in this case, after the resize has occurred). ***CS_GetAction*** returns a value of 5 (user resized a banner). You can then call ***CS_GetResizBanr*** ([page 57](#)) to determine which banner was resized.

CalendarSet v3 requires the Macintosh Drag and Drop extension to perform event and banner dragging and banner re-sizing. However, other CalendarSet features do not require the Macintosh Drag and Drop extension unless specified otherwise. If the Macintosh Drag and Drop extension is not present in the Extensions folder, CalendarSet will ignore attempts to drag events or banners, or resize

banners. The Macintosh Drag and Drop Manager is included in System 7.5, and can be installed into earlier versions of System 7.x. The Macintosh Drag and Drop extension cannot be installed into System 6.

The CalendarSet disk has the Macintosh Drag and Drop extension included, and you are licensed to install it as part of your database application. [See “Installation: Plug-In \(Macintosh\)” on page 9.](#)

Word Wrap

CalendarSet will optionally word wrap event text to fit within a day’s width, rather than truncating text which doesn’t fit. This is controlled by the *ShowWordWrap* parameter of ***CS_SetEventOpts*** ([page 42](#)) to 1.

Event Markers

To distinguish two or more events from each other within a day, each event can be optionally marked using a special character. The parameter *EventMarker* can be set to 0, 1 or 2 to show no mark, a bullet, or a dash respectively. If 0 (no mark) is chosen (which is the default), you can choose a marker of choice by setting the first character of the event text passed in using ***CS_SetArray*** ([page 35](#)) to the desired character followed by a space. Possible marker candidates from the Geneva font are “• - ð Þ —.”

No word wrapping will be performed for banners.

Hiding Days

You can hide one or more days in a week using ***CS_HideDays*** ([page 46](#)).

Resizable Calendar Areas on a Form

Whether using 4D v6 or an earlier version of 4D, you can configure CalendarSet objects to be resizable on a 4D form (layout). In 4Dv6, you just have to make the appropriate changes to the CalendarSet object’s properties, the layout’s properties and choose the proper window type and you’re done. Please refer to the 4th Dimension documentation for specific details.

When using an earlier version of 4D, there are some new commands in CalendarSet to facilitate resizing. The syntax of these commands are described below.

To Create a Resizable Calendar Object

- 1 Create a CalendarSet area on a layout. Make sure that the area extends well beyond (in the 1200 to 1300 pixel range) the right and bottom sides of the layout.
- 2 In the Before phase, call `CS_SetSizeOpts` ([page 51](#)) to allow resizing and to specify both the min/max width/height of the area.
- 3 Make certain that the window type used does not contain a size (grow) box. Types 4 or 12 are recommended. If a window type with a size box is chosen, CalendarSet cannot limit the size of the window to the minimum or maximum settings.
- 4 Within the script of the CalendarSet area, call `CS_GetAction` ([page 53](#)). If the action is 6, Window Resized, call `CS_DoResize` ([page 59](#)). This command allows CalendarSet to change the window size to the what the user selected via the size box.

Commands

CS_Register

(registrationKey:S) → resultCode:L

Parameter	Type	Description
→ registrationKey	string	Registration key
← resultCode	longint	Result code

CS_Register is used to register the CalendarSet plug-in for developer or deployment use. You must call **CS_Register** with a valid registration key; otherwise CalendarSet will operate in demonstration mode.

Without a valid registration key, CalendarSet will operate in demonstration mode during 20 minutes.

Like all e-Node plug-ins, CalendarSet offers six different license types. There are no such things as MacOS vs Windows or Development vs Deployment:

- **Single user license.** This license allows development (interpreted mode) or deployment (interpreted or compiled mode) on 4D Standalone or Runtime. Since the registration key is linked to a specific 4D license, you need to provide the number returned by the 4D command **GET SERIAL INFORMATION** (first parameter). A new license will be provided for free at any time if you change your 4D version and/or get a new 4D registration key.
- **Small server.** This license allows development (interpreted mode) or deployment (interpreted or compiled mode) on 4D Server up to 10 users. The registration key is linked to your 4D Server license just as above.
- **Medium server.** This license allows development (interpreted mode) or deployment (interpreted or compiled mode) on 4D Server up with 11 to 20 users. The registration key is linked to your 4D Server license just as above.

Configuration Commands

- **Large server.** This license allows development (interpreted mode) or deployment (interpreted or compiled mode) on 4D Server over 20 users. The registration key is linked to your 4D Server license just as above.
- **Unlimited Single User.** This license allows development (interpreted mode) or deployment (interpreted or compiled mode) on as many 4D Standalone, Runtime or Engine copies that run your 4D application(s). This is a yearly license, which expires one month after the date when it is to be renewed. The expiration only affects interpreted mode. **Compiled applications using an obsolete license will never expire.**
- **Unlimited OEM.** This license allows development (interpreted mode) or deployment (interpreted or compiled mode) on as many 4D Server (of any number of users), 4D Standalone, Runtime or Engine copies that run your 4D application(s). This is a yearly license, which expires one month after the date when it is to be renewed. The expiration only affects interpreted mode. **Compiled applications using an obsolete license will never expire.**

A 4D database used to retrieve your 4D serial information is available from the following link:

<http://www.e-node.net/ftp/GetSerialInfo>

The registration system has been modified in version 3.6. Only one registration key is now required.

registrationKey — Pass the registration key to register your copy of CalendarSet. Only one registration key is required. The key is either linked to the 4D or 4D Server serial number, or to the name of the company/developer, depending on the license type.

resultCode — This will return a value of 1 if the registration key is valid and a value of 0 if the registration key is invalid. You should verify the correctness of the registration key by tracing over the call to **CS_Register** and examining **resultCode**.

Example:

```
C_LONGINT($result)
$result:=CS_Register("Place your registration key here")
If($result#1) `error
    ALERT("CalendarSet could not be registered:"+String($result))
End if
```

_CS_Area

_CS_Area is the command used to identify the CalendarSet plug-in area when you create a plug-in area object on a layout. This command is only used in the object definition for an CalendarSet object, and should never be used as a command in a method.

CS_SetRange

CS_SetRange(*area name*;StartDate;EndDate)

Parameter	Type	Description
<i>area name</i>	longint	name of CalendarSet object on layout
<i>StartDate</i>	date	the first date to be displayed in the calendar
<i>EndDate</i>	date	the last date to be displayed in the calendar

CS_SetRange tells CalendarSet the date range to display. The date range will default to the current month if this command is not used.

StartDate — Date. This value will set the first date for the Calendar.

EndDate — Date. This value will set the last date for the Calendar.

Pass a value of !00/00/00! for *EndDate* to have CalendarSet set *EndDate* to be the last day of the month specified in *StartDate*.

The current selection of cells is lost when this command is issued. To maintain the selection, use **CS_GetSelect** ([page 52](#)) prior to calling **CS_SetRange**, and **CS_SetSelect** ([page 34](#)) afterward.

Examples:

```
`set the calendar to display the month of March, 1995
CS_SetRange (eCalendar;!03/01/93!;!00/00/00!)
```

```
`set the calendar to display the first week of June, 1995
CS_SetRange (eCalendar;!06/01/93!;!06/07/95!)
```

CS_SetSelect

CS_SetSelect(*area name*;StartDate;EndDate;Contiguous;DaysArray)

Parameter	Type	Description
<i>area name</i>	longint	name of CalendarSet object on layout
<i>StartDate</i>	date	the start date of the selection
<i>EndDate</i>	date	the end date of the selection
<i>Contiguous</i>	integer	1 if selection is contiguous, 0 if not
<i>DaysArray</i>	date array	array of currently selected items

CS_SetSelect sets the current selection of the calendar. This is the opposite action of **CS_GetSelect** ([page 52](#)).

StartDate — Date. This parameter specifies the first day to highlight.

EndDate — Date. This parameter specifies the last day to highlight. All days between and including *StartDate* and *EndDate* will be highlighted if *Contiguous* is 1.

Contiguous — Integer.

- 1 the current selection is set to the range *StartDate* to *EndDate*,
- 0 the current selection is set to be each of the days that are specified in *DaysArray*

DaysArray — Date array. This parameter is used to past a list of discontinuous days you wish to select (highlight).

Example:

```

`highlight the current date
ARRAY DATE(aDate;0)
C_DATE(vStart;vEnd)
vStart:=Current date
vEnd:=vStart
CS_SetSelect (eCalendar;vStart;vEnd;1;aDate)
`highlight the first three days of March 1995
CS_SetSelect(eCalendar;!03/01/95;!03/03/95!;1;aDate)
`highlight the 1st and 15th of June 1995
ARRAY DATE(aDate;2)
aDate{1}:=!05/01/95!
aDate{2}:=!05/15/95!
CS_SetSelect(eCalendar;!00/00/00;!00/00/00!;0;aDate)

```

CS_SetArray

CS_SetArray(*area name*;DateArray;TextArray; FontArray; SizeArray; StyleArray; ColorArray)

Parameter	Type	Description
<i>area name</i>	longint	name of CalendarSet object on layout
<i>DateArray</i>	string	dates of each item
<i>TextArray</i>	string	text of each item to display (name of text or string array)
<i>FontArray</i>	string	font to use for each item (name of text or string array)
<i>SizeArray</i>	string	size of each item (name of numeric array)
<i>StyleArray</i>	string	style of each item (name of numeric array)
<i>ColorArray</i>	string	color of each item (name of numeric array)

CS_SetArray is used to specify the array that is used to display events in each of the days. The arrays passed to this command are typically loaded using the 4D command **SELECTION TO ARRAY**, with the values in records containing event or other date-based information.

Each of the arrays can be thought of as a column of a table, and

each row of that table contains the array elements which correspond to an event record. For a particular record, there is information for the date of the event, the text, and font, size, style, and color.

If the arrays are changed after calling **CS_SetArray**, you will need to call **Area_Refresh** ([page 88](#)) to force CalendarSet to show the changes.

Note: modifying the event arrays doesn't require reissuing CS_SetArray, although you must call Area_Refresh ([page 88](#)).

This command is optional (although usually you will use it), and is only used to display text events on one or more days.

Note: The array parameters are actually strings and not the arrays themselves.

DateArray — String (name of a date array). Used to define which days you want the event to be displayed on.

TextArray — String (name of a string or text Array). Used to define the text of the events to be displayed.

FontArray — String (name of a string or text array). The referenced array contains the fonts to display each corresponding event. To use the default information for font characteristics, pass an empty string for the **FontArray** parameter.

SizeArray — String (name of a numeric array). The referenced array is used to specify the font size of the text for each corresponding event. Pass an empty string ("") to use the default size for all events.

StyleArray — String(name of a numeric array). The referenced array is used to specify the style (plain, bold, italic, etc.) of the text for each corresponding event. Pass an empty string ("") to use the default style for all events.

ColorArray — String (name of a numeric array). Each item in the color array allows you to specify either the index into the 4th Dimension color palette (positive numbers) or one of the hardcoded numbers specified in the color table shown below. Pass an empty string ("") to use the default color for all events.

Table 1:

Color	Value
Black	-1

Table 1:

Color	Value
White	-2
Red	-3
Green	-4
Blue	-5
Cyan	-6
Magenta	-7
Yellow	-8
4D Colors	1 to 256

The 4D color palette is a 16 by 16 grid. To determine a color's value, you can locate the color's position on the color grid in the Design environment (the Color submenu which is available in the Layout and Procedure editors), and count the number of rows down and columns across. The equation is

$$\text{ColorValue} = ((\text{RowNumber} - 1) \times 16) + \text{ColumnNumber}$$

Example:

```

`display all items for this month from the appointments file
C_DATE(vDay1ThisMth;vDay1NextMth)
C_INTEGER($ThisMonth)
$ThisMonth:=Month of(Current date)
`See "Util_SetDate" on pag e87
Util_SetDate(vDay1ThisMth;$ThisMonth;1;Year of(Current date))
If ($ThisMonth<12)
    Util_SetDate(vDay1NextMth;$ThisMonth+1;1;Year of(Current date))
Else
    Util_SetDate(vDay1NextMth;1;1;Year of(Current date)+1)
End if
query([Appts];[Appts]Appt Date>=vDay1ThisMth;*) `find the appointments for
    this month
query([Appts]; & ;[Appts]Appt Date<vDay1NextMth)
`now load the appointment data into arrays
`Note that each appointment item has a field for the date, item text,
`font, size, style, and color. This makes loading the data and displaying it
`very easy to accomplish.
SELECTION TO ARRAY([Appts]Appt Date;aDates;[Appts]Item;alt-
    ems;[Appts]Item
    Font;aFonts;[Appts]Size;aSizes;[Appts]Styles;aStyles;[Appts]Col-
    or;aColors)
`and display the data in the CalendarSet plug-in area named "eMonth"
CS_SetArray (eMonth;"aDates";"altems";"aFonts";"aSizes";"aStyles";"aColors")

```

CS_SetBanrArray

CS_SetBanrArray(*area name*; *StartDateArray*; *EndDateArray*; *TextArray*; *FontArray*; *SizeArray*; *StyleArray*; *ForeColorArray*; *BackColorArray*)

Parameter	Type	Description
<i>area name</i>	longint	name of CalendarSet object on layout
<i>StartDateArray</i>	string	start dates for each banner
<i>EndDateArray</i>	string	end dates for each banner
<i>TextArray</i>	string	text for each banner
<i>FontArray</i>	string	font for each banner
<i>SizeArray</i>	string	size for each banner
<i>StyleArray</i>	string	style for each banner
<i>ForeColorArray</i>	string	foreground color for each banner
<i>BackColorArray</i>	string	background color for each banner

CS_SetBanrArray allows you to specify the arrays that are used to draw the banners that are displayed in the calendar. *Note that the arrays are passed in quotes, which is why the parameter type is string.* The size, style, foreground color, and background color arrays are of type integer.

*Note that modifying the banner arrays doesn't require reissuing **CS_SetIBanrArray**, although you must call **Area_Refresh** ([page 88](#)).*

StartDateArray — String (name of a date array). The referenced array is used to define the start date for each banner.

EndDateArray — String (name of a date array). The referenced array is used to define the end date for each banner.

TextArray — String (name of a string or text Array). Used to define the text of the banners to be displayed.

FontArray — String (name of a string or text array). The referenced array contains the fonts to display each corresponding banner. To use the default information for font characteristics, pass an empty string for the *FontArray* parameter.

SizeArray — String (name of a numeric array). The referenced array is used to specify the font size of the text for each corresponding banner.

StyleArray — String(name of a numeric array). The referenced array is used to specify the style (plain, bold, italic, etc.) of the text for each corresponding banner.

ForeColorArray — String (name of a numeric array). The referenced array is used to specify the foreground color for each banner. Each item in the color array allows you to specify either the index into the 4th Dimension color palette (positive numbers) or one of the hard-coded numbers specified in the color table shown below.

BackColorArray — String (name of a numeric array). The referenced array is used to specify the background color for each banner. Each item in the color array allows you to specify either the index into the 4th Dimension color palette (positive numbers) or one of the hard-coded numbers specified in the color table shown below.

Table 2:

Color	Value
Black	-1
White	-2
Red	-3
Green	-4
Blue	-5
Cyan	-6
Magenta	-7
Yellow	-8
4D Colors	1 to 256

The 4D color palette is a 16 by 16 grid. To determine a color's value, you can locate the color's position on the color grid in the Design environment (the Color submenu which is available in the Layout and Procedure editors), and count the number of rows down and columns across. The equation is

$$\text{ColorValue} = ((\text{RowNumber} - 1) \times 16) + \text{ColumnNumber}$$

Example:

CS_SetBanrArray (eMonth;"aStDates";"aEnDates";"altem-
Texts";"aFonts";"aSizes"; "aStyles";"aForeColors";"aBackColors")

CS_SetIconArray

CS_SetIconArray(*area name*; *DateArray*; *IconArray*; *SizeArray*)

Parameter	Type	Description
<i>area name</i>	longint	name of CalendarSet object on layout
<i>DateArray</i>	string	date for display of each icon
<i>IconArray</i>	string	resource id numbers for each icon
<i>SizeArray</i>	string	size for each icon

CS_SetIconArray is used to specify the array that is used to display icons in each of the days.

Note: DateArray, IconArray, and SizeArray are the names of the arrays, enclosed in quotes.

Note: modifying the icon arrays doesn't require reissuing CS_SetIconArray, although you must call Area_Refresh (page 88).

DateArray — String. This parameter is the name of a date array, and is used to define which days that you want icons displayed on.

IconArray — String. This parameter is the name of an integer array and specifies the icon resource ids of the icons to display. The icons specified in *IconArray* can be stored either in the Proc.Ext file or in the structure file of the application. There can be, at most, one icon per day in a calendar.

SizeArray — String. This parameter is the name of an integer array and specifies the size of the icons to be displayed. The following values are allowed:

- 32** Large Icons (32 x 32) — resource types icl8, icl4, or ICN#
- 16** Small Icons (16 x 16) — resource types ics8, ics4, or ics#
- p0** Best Size — CalendarSet will draw the size that fits best

You must use a utility application such as ResEdit or Resourcer to add or remove a file's icon resources. Always make a backup of the file prior to opening it with one of these utilities.

Note: When adding, modifying, or deleting icons from either file, use the 4D Customizer Plus utility to update the compatibility (4D4D) resource. This is required when running your database with 4D Server, to ensure that a new copy of the .Res file is downloaded to each client Macintosh.

Example:

SELECTION TO ARRAY([Dates]Appt Date;alconDates;[Dates]Icon ID;alconIDs;[Dates]Icon Size;alconSizes)

CS_SetIconArray(eArea;"alconDates";"alconIDs";"alconSizes")

`add an icon to the date 4/12/93 on the displayed calendar eCal
`the icon info is in the arrays alconDates, alconIDs, and alconSizes

\$Size:=**Size of array**(alconDates)+1

ARRAY DATE(alconDates;\$Size)

ARRAY INTEGER(alconIDs;\$Size)

ARRAY INTEGER(alconSizes;\$Size)

alconDates{\$Size}:=**vlconDate**

alconIDs{\$Size}:=**vlconID**

alconSizes{\$Size}:=**vlconSize**

Area_Refresh (eCalendar)

CS_Options

CS_Options(AreaName;DayHighlightMode;DaySelectionType;UnusedInfo;MonthPrefix;
FirstDayOfWeek;ExtendFrame)

Parameter	Type	Description
<i>AreaName</i>	longint	name of CalendarSet object on layout
<i>DayHighlightMode</i>	integer	type of highlighting to use for day selection
<i>DaySelectionType</i>	integer	type of selection allowed
<i>UnusedInfo</i>	integer	how to display unused cells
<i>MonthPrefix</i>	integer	display the month name for first day of month
<i>FirstDayOfWeek</i>	integer	which day starts a week
<i>ExtendFrame</i>	integer	include headers within enclosing frame

CS_Options is used to control several CalendarSet options. Each parameter is an integer, and is defined as follows:

DayHighlightMode — Integer, 0, 1, or 2. Type of highlighting to use for selected days (cells). There are three possible values:

- 0 No Highlight
- 1 Highlight Entire Day's Cell
- 2 Highlight Number only

DaySelectionType — Integer, 0, 1, or 2. Controls how days are selected. There are three possible values:

- 0 Single day only (default)
- 1 Multiple days selectable
- 2 Discontiguous day selectable

UnusedInfo — Integer, 0, 1, or 2. Defines how unused days will be displayed. Unused days are those outside of the date range defined

with **CS_SetRange** ([page 34](#)). There are three possible values:

- 0 Nothing displayed
- 1 Cell is grayed out
- 2 Cell is grayed and day number and banners are displayed (default)

MonthPrefix — Integer, 0 or 1. A value of 1 will display the month name for the first day of every month. The default is zero.

FirstDayOfWeek — Integer, 0 - 6. Controls the day of the week on which a week will begin.

- 0 Sunday (default)
- 1 Monday
- 2 Tuesday
- 3 Wednesday
- 4 Thursday
- 5 Friday
- 6 Saturday

ExtendFrame — Integer, 0 or 1.

- 1 the frame that is drawn around the CalendarSet area is extended to include the header labels representing the days of the week
- 0 CalendarSet will not extend the frame (default)

Examples:

`highlight number only, multiple cells (contiguous), grayed cells with info, month prefix, sunday start, extend frame

CS_Options (eCalendar;2;1;2;1;0;1)

`highlight entire cell, multiple cells (discontiguous), unused cells blank, no month prefix, monday start, don't extend frame

CS_Options (eCalendar;1;2;0;0;1;0)

CS_SetEventOpts

CS_SetEventOpts(AreaName; AllowEventSelect; ShowWordWrap; EventMarker)

Parameter	Type	Description
<i>AreaName</i>	longint	name of CalendarSet object on layout
<i>AllowEventSelect</i>	integer	allow selection of events
<i>ShowWordWrap</i>	integer	word-wrap event text
<i>EventMarker</i>	integer	character to mark each event

CS_SetEventOpts is used to control whether events are selectable, word-wrapped, and the marker character to use (if any) for each event.

AllowEventSelect — Integer, 0 or 1. Use this parameter to allow or disallow the selection of events.

- 0 no event selection allowed (default)
- 1 single event select

ShowWordWrap — Integer, 0 or 1. Use this parameter to word-wrap the text of an event to fit within a day. If word-wrapping is not enabled, then event text will be truncated on the right edge of each day.

- 0 no word wrapping will be performed (default)
- 1 indicates that word wrapping will be performed on all events

EventMarker — Integer. 0, 1 or 2. This parameter is used to control the symbol that is shown to the left of the event text. The symbol is used to help the user distinguish different events (useful when *ShowWordWrap* is 1).

- 0 no marker (default)
- 1 bullet
- 2 dash

Examples:

```
`allow event selection, word-wrap event text, mark events with bullet  
CS_SetEventOpts(eCal;1;1;1)  
`no event selection, word-wrap event text, mark events with dash  
CS_SetEventOpts(eCal;0;1;2)
```

CS_SetBanrOpts

CS_SetBanrOpts(AreaName; AllowBannerSelect; AllowBannerResize)

Parameter	Type	Description
<i>AreaName</i>	longint	name of CalendarSet object on layout
<i>AllowBannerSelect</i>	integer	allow selection of banners
<i>AllowBannerResize</i>	integer	allow resizing of banners

CS_SetBanrOpts is used to control whether banners are selectable and/or resizable. You can detect the selection of a banner using **CS_GetEvtSelect** ([page 54](#)). You can detect that a banner has been resized using **CS_GetResizBanr** ([page 57](#)).

AllowBannerSelect — Integer, 0 or 1. Use this parameter to allow or disallow the selection of banners.

- 0 no banner selection allowed (default)
- 1 single banner select

AllowBannerResize — Integer, 0 or 1 Use this parameter to allow or disallow the resizing of banners.

- 0 banner resizing not allowed (default)
- 1 the user will be able to resize banners by clicking on either of the banner's flags and dragging to a new date

Examples:

```
`allow banner selection, allow banner resizing
CS_SetBanrOpts(eCal;1;1)
`no banner selection, no banner resizing
CS_SetBanrOpts(eCal;0;0)
```

CS_SetCalColor

CS_SetCalColor(AreaName; CalForeColor; CalBackColor)

Parameter	Type	Description
<i>AreaName</i>	longint	name of CalendarSet object on layout
<i>CalForeColor</i>	integer	color of calendar unused days
<i>CalBackColor</i>	integer	color of calendar background

CS_SetCalColor is used to set the foreground and background color of the calendar. The unused days (those outside the date range defined using **CS_SetRange** ([page 34](#))) are drawn in the foreground color. The entire background of the calendar is shown in the background color.

CalForeColor — Integer. Refer to the color table shown below. The default is black.

CalBackColor — Integer. Refer to the color table shown below. The default is white.

Table 3:

Color	Value
Black	-1
White	-2
Red	-3
Green	-4
Blue	-5
Cyan	-6
Magenta	-7

Table 3:

Color	Value
Yellow	-8
4D Colors	1 to 256

The 4D color palette is a 16 by 16 grid. To determine a color's value, you can locate the color's position on the color grid in the Design environment (the Color submenu which is available in the Layout and Procedure editors), and count the number of rows down and columns across. The equation is

$$\text{ColorValue} = ((\text{RowNumber} - 1) \times 16) + \text{ColumnNumber}$$

Example:

```
`unused days color blue, background color very light gray
CS_SetCalColor(eCal;-5;241)
```

CS_SetEvtSelect

CS_SetEvtSelect(AreaName; EventType; EventIndex)

Parameter	Type	Description
<i>AreaName</i>	longint	name of CalendarSet object on layout
<i>EventType</i>	integer	event or banner
<i>EventIndex</i>	integer	array element corresponding to item being selected

CS_SetEvtSelect is used to procedurally select events or banners.

EventType — Integer, 1 or 2. Use this parameter to specify if you are selecting events or banners.

- 1 event
- 2 banner

EventIndex — Integer. Index within the array that was passed to CalendarSet via either ***CS_SetArray*** ([page 35](#)) or ***CS_SetBanrArray*** ([page 38](#)).

Use ***CS_GetEvtSelect*** ([page 54](#)) to determine what event or banner was selected by the user.

Examples:

```
`set the first event to be selected
CS_SetEvtSelect(eCal;1;1)
`set the third event to be selected
CS_SetEvtSelect(eCal;1;3)
`set the fourth banner to be selected
```

CS_SetEvtSelect(eCal;2;4)

CS_HideDays

CS_HideDays(area name; HideSunday; HideMonday; HideTuesday; HideWednesday; HideThursday; HideFriday; HideSaturday)

Parameter	Type	Description
<i>area name</i>	longint	name of CalendarSet object on layout
<i>HideSunday</i>	integer	don't display Sunday
<i>HideMonday</i>	integer	don't display Monday
<i>HideTuesday</i>	integer	don't display Tuesday
<i>HideWednesday</i>	integer	don't display Wednesday
<i>HideThursday</i>	integer	don't display Thursday
<i>HideFriday</i>	integer	don't display Friday
<i>HideSaturday</i>	integer	don't display Saturday

CS_HideDays allows you to procedurally hide any of the days on the calendar. This is most useful for hiding the weekend days to only show a weekday calendar.

A value of one for any day will hide that day, a value of zero will show the day.

The default for all days is zero (all days are visible by default).

Examples:

```

`hide saturday and sunday
CS_HideDays (eCal; 1; 0; 0; 0; 0; 0; 1)
`hide sunday only
CS_HideDays (eCal; 1; 0; 0; 0; 0; 0; 0)
`show saturday and sunday only
CS_HideDays (eCal; 0; 1; 1; 1; 1; 1; 0)
`show only the first day in May, 1993
CS_SetRange (eCal;Current date;Current date) `set active range to current date
only
C_INTEGER(v1;v2;v3;v4;v5;v6;v7;vDayNum) `declare variables for use below
For($i;1;7) `initialize the variables to hide all days
    $TempPtr:=Get pointer("v"+String($i))
    $TempPtr»:=1
End for
vDayNum:=Day number(Current date)
$TempPtr:=Get pointer("v"+String(vDayNum))
$TempPtr»:=0 `don't hide today
CS_HideDays (eCal;v1;v2;v3;v4;v5;v6;v7)

```

CS_FontDefaults

CS_FontDefaults(area name; Selector; Font; Size; Style; ForeColor; BackColor)

Parameter	Type	Description
<i>area name</i>	longint	name of CalendarSet object on layout
<i>Selector</i>	integer	attribute identifier
<i>Font</i>	string	font name
<i>Size</i>	integer	font size
<i>Style</i>	integer	font style
<i>ForeColor</i>	integer	foreground color
<i>BackColor</i>	integer	background color (banners only)

CS_FontDefaults allows you to specify the default font characteristics for various items in the calendar.

Selector — Integer. Specifies which attribute that you want to set. The values possible for *Selector* are:

Table 4:

Value	Attribute	Default values
1	day numbers	Geneva, 9, bold, black
2	text that appears in each cell	Geneva, 9, plain, black
3	day names that appear at the top of the calendar	Geneva, 9, bold, black
4	banners	Geneva, 9, plain, black, white (background color)
5	current date day number	Geneva, 9, bold, black

Font — String. Use this parameter to specify the font for the selected attribute.

Note: the default font is actually the application font, which is usually Geneva in the US. A different font may be used with non-US versions of the Macintosh operating system.

Size — Integer. Use this parameter to set the font size of the selected attribute. Pass an empty string ("") to use the default font size for the specified attribute.

Style — Integer. Use this parameter to set the font style of the selected attribute. Pass an empty string ("") to use the default font style for the specified attribute.

ForeColor — Integer. Use this parameter to set the foreground color of the selected attribute. Refer to the color table below for possible values. Pass an empty string (“”) to use the default foreground color for the specified attribute.

BackColor — Integer. Use this parameter to set the background color of the selected attribute. Refer to the color table below for possible values. Pass an empty string (“”) to use the default background color for the specified attribute.

Table 5:

Color	Value
Black	-1
White	-2
Red	-3
Green	-4
Blue	-5
Cyan	-6
Magenta	-7
Yellow	-8
4D Colors	1 to 256

The 4D color palette is a 16 by 16 grid. To determine a color's value, you can locate the color's position on the color grid in the Design environment (the Color submenu which is available in the Layout and Procedure editors), and count the number of rows down and columns across. The equation is

$$\text{ColorValue} = ((\text{RowNumber} - 1) \times 16) + \text{ColumnNumber}$$

The font settings for day numbers, day names, and banners can be overridden on an item by item basis using the routines:

CS_SetDayStyle ([page 49](#)), ***CS_SetDayStyleB*** ([page 50](#)), ***CS_SetArray*** ([page 35](#)), and ***CS_SetBanrArray*** ([page 38](#)).

Examples:

```
`show day number in New York 10 point plain type, black color
CS_FontDefaults (eCal; 1; "New York"; 10; 0; 0)
`show day names in Helvetica 9point bold type, black color
CS_FontDefaults (eCal; 3; "Helvetica"; 9; 1; 0)
```

CS_SetDayStyle

CS_SetDayStyle(*area name*; *TargetDate*; *DayFont*; *DaySize*; *DayStyle*; *DayColor*; *ClearOthers*)

Parameter	Type	Description
<i>area name</i>	longint	name of CalendarSet object on layout
<i>TargetDate</i>	date	date to change the font info
<i>DayFont</i>	string	font name for the day number
<i>DaySize</i>	integer	font size for the day number
<i>DayStyle</i>	integer	font style for the day number
<i>DayColor</i>	integer	font color for the day number
<i>ClearOthers</i>	integer	clear custom info for all other days

CS_SetDayStyle allows you to change the font and style of the day number for any of the days in the calendar. You specify the date to be changed, as well as the font, size, style, and color for that date.

TargetDate — Date. This parameter specifies the date that the command will act on. The *TargetDate* does not have to be part of the currently displayed date range.

DayFont — String. This parameter specifies the font to use for *TargetDate*.

DaySize — Integer. This parameter specifies the font size to use for *TargetDate*.

DayStyle — Integer. This parameter specifies the font style to use for *TargetDate*.

DayColor — Integer. This parameter specifies the foreground color to use for *TargetDate*. Refer to the table below for possible color values.

Table 6:

Color	Value
Black	-1
White	-2
Red	-3
Green	-4
Blue	-5
Cyan	-6

Table 6:

Color	Value
Magenta	-7
Yellow	-8
4D Colors	1 to 256

The 4D color palette is a 16 by 16 grid. To determine a color's value, you can locate the color's position on the color grid in the Design environment (the Color submenu which is available in the Layout and Procedure editors), and count the number of rows down and columns across. The equation is

$$\text{ColorValue} = ((\text{RowNumber} - 1) \times 16) + \text{ColumnNumber}$$

ClearOthers — Integer. Use this parameter to specify whether or not you want the special font info cleared for all other days in the calendar. This is useful if you just want to have one day have special font info.

Example:

```
`display the current date in Times 12 point bold, black color. Set all
`other days to the default style.
$Today:=Current date
CS_SetDayStyle (eCalArea; $Today; "Times"; 12; 1; 0; 1)
```

CS_SetDayStyleB

CS_SetDayStyleB(*area name*; *DateArray*; *DayFont*; *DaySize*; *DayFace*; *DayColor*; *ClearOthers*)

Parameter	Type	Description
<i>area name</i>	longint	name of CalendarSet object on layout
<i>DateArray</i>	date array	dates to change the font info
<i>DayFont</i>	string	font name for the day number
<i>DaySize</i>	integer	font size for the day number
<i>DayFace</i>	integer	font style for the day number
<i>DayColor</i>	integer	font color for the day number
<i>ClearOthers</i>	integer	clear custom info for all other days

CS_SetDayStyleB is identical to **CS_SetDayStyle** ([page 49](#)) except that it accepts a date array (instead of a date variable) as the second argument. It applies the specified style to all days in the array.

Example:

```
`show all days with appoints (contained in array aDates) as
```

``Palatino 10 point bold italic, black color. Don't clear other day formatting.
CS_SetDayStyleB (eCalendar; aDates; "Palatino"; 10; 3; 0; 0)`

CS_SetSizeOpts

CS_SetSizeOpts (*AreaName*; *AllowResize*; *MinWidth*; *MinHeight*; *MaxWidth*; *MaxHeight*)

Parameter	Type	Description
<i>AreaName</i>	longint	name of CalendarSet object on layout
<i>AllowResize</i>	integer	allow CalendarSet to be resized
<i>MinWidth</i>	integer	minimum width of the CalendarSet area
<i>MinHeight</i>	integer	minimum height of the CalendarSet area
<i>MaxWidth</i>	integer	maximum width of the CalendarSet area
<i>MaxHeight</i>	integer	maximum height of the CalendarSet area

CS_SetSizeOpts will enable resizing of the CalendarSet area when using 4D v3. This command is not necessary when using 4D v6. [Please read the section “Resizable Calendar Areas on a Form” on page 31 for more information.](#)

AllowResize —integer. This parameter specifies whether the CalendarSet object specified by *AreaName* can be resized.

- 0 do not allow resizing (default)
- 1 allow resizing

MinWidth —integer. This parameter specifies the minimum width of the CalendarSet object, in pixels.

MaxWidth — integer. This parameter specifies the maximum width of the CalendarSet object, in pixels.

MinHeight —integer. This parameter specifies the minimum height of the CalendarSet object, in pixels.

MaxHeight —integer. This parameter specifies the maximum height of the CalendarSet object, in pixels.

Note: the minimum and maximum values apply to the CalendarSet object — not to the window.

User Action Commands

Responding to User Actions on a CalendarSet Object

User interaction with a CalendarSet object is handled in the **During** phase. Once again, the CalendarSet object script will contain the procedures to detect user actions, such as selection, double-clicks, etc. You may also use certain CalendarSet commands in other scripts or procedures, such as a query button script which changes the arrays currently displayed in a CalendarSet object. In such a script, if you modify the arrays (or the CalendarSet object's configuration) in any way, you'll need to use **Area_Refresh** ([page 88](#)) to update the CalendarSet object.

The user's action is communicated to you using **CS_GetAction** ([page 53](#)) whenever the CalendarSet script (or for an plug-in window, the callback procedure) is executed — i.e., the user has clicked on the CalendarSet area. **CS_GetAction** returns either a day was selected, or an event was selected.

If an event was selected, you can then call **CS_GetEvtSelect** ([page 54](#)) to determine the event that is selected. **CS_LastClick** ([page 56](#)) is available to obtain more information about the click itself, such as whether the click was a double-click, and what modifier keys were used.

You can select an event procedurally using **CS_SetEvtSelect** ([page 45](#)), which is similar in syntax to **CS_GetEvtSelect** ([page 54](#)).

You can respond to a window resize action using **CS_DoResize** ([page 59](#)), if running 4D v3. *Please read the section "[Resizable Calendar Areas on a Form](#)" on page 31 for more information.*

CS_GetSelect

CS_GetSelect(*area name;StartDate;EndDate;Contiguous;DaysArray*)

Parameter	Type	Description
<i>area name</i>	longint	name of CalendarSet object on layout
<i>StartDate</i>	date	the start date of the selection
<i>EndDate</i>	date	the end date of the selection
<i>Contiguous</i>	integer	are selected dates adjacent to each other
<i>DaysArray</i>	date array	array of currently selected items

CS_GetSelect will return the days that are currently highlighted in

the calendar.

StartDate — Date. This value is the first day the is currently selected. A value of !00/00/00! is returned if no days are selected.

EndDate — Date. This value is the last day that is currently selected. A value of !00/00/00! is returned if no days are selected.

Contiguous — Integer. This value indicates whether or not all of the cells that are selected are contiguous (located adjacent to each other).

1 contiguous
0 non-contiguous

DaysArray — Date array. This array will be filled with the dates of each of the selected items in the calendar, if the selected dates are discontiguous (not located adjacent to each other).

Use **CS_SetSelect** ([page 34](#)) to procedurally highlight one or more days.

Example:

```

`get the selection of days for the eCalendar object
`use the selected dates as query criteria for a query
`of the [Followup] file
C_DATE(vStart;vStop)
ARRAY DATE(aDates;0)
C_INTEGER(vContig)
CS_GetSelect (eCalendar;vStart;vStop;vContig;aDates) `get the selected days
if(vStart#!00/00/00!) `make sure at least one day is selected
  query([Followup];[Followup]Followup Date=aDates{1};*) `start the built query
  for($i;2;Size of array(aDates)) `step thru each element of the date array
    query([Followup]; | ;[Followup]Followup Date=aDates{$i};*)
  end for
  query([Followup]) `kick off the built query
end if

```

CS_GetAction

CS_GetAction (*AreaName*;*ActionCode*)

Parameter	Type	Description
<i>AreaName</i>	longint	name of CalendarSet object on layout
<i>ActionCode</i>	integer	type of user action

This routine is called from the CalendarSet script (or for a Calendar-Set plug-in window, from the *ProcToExecute* — [See “CS_SetCallback” on page 58](#)) in order to determine the last Calen-

darSet action. *Please read the section [“Responding to User Actions on a CalendarSet Object” on page 52 for more information.](#)*

ActionCode — Integer, 1 - 6. This parameter is returned with a value indicating what action the user made on the CalendarSet object.

Possible values are:

- 1 user click on day (no event selected)
- 2 user closed the window (valid only for a CalendarSet plug-in window)
- 3 user clicked on an event (event or banner)
- 4 user dragged an event (event or banner)
- 5 user resized a banner
- 6 user resized the window

Example:

```
`eCalendar script, detect a double-click on a day. If no modifier keys  
`depressed, add an event. If command key depressed, modify an event
```

Case of

```
:(During)  
  CS_LastClick (eCalendar;vDate;vDC;vMod)  
  If(vDC=1) `was it a double-click?  
    If((vMod\256)%2=1) `command key?  
      MOD CAL EVT (vDate)  
    Else  
      ADD CAL EVT (vDate)  
    End if  
  End if  
End case
```

CS_GetEvtSelect

CS_GetEvtSelect(AreaName; EventType; EventIndex)

Parameter	Type	Description
AreaName	longint	name of CalendarSet object on layout
EventType	integer	event or banner selected
EventIndex	integer	array element corresponding to selected item

CS_GetEvtSelect is used to determine what event was selected by the user. You will usually call *CS_GetAction* ([page 53](#)) prior to this command.

EventType — Integer, 1 or 2. The value returned in *EventType* indicates whether the selected item is an event or banner.

- 1 event
- 2 banner

EventIndex — Integer. This value is an index within the array that was passed to CalendarSet via either *CS_SetArray* ([page 35](#)) or

CS_SetBanrArray ([page 38](#)).

Use **CS_SetEvtSelect** ([page 45](#)) to procedurally select an event or banner.

Example:

```

C_LONGINT(vAction;vType;vIndex)
if(During)
  CS_GetAction(eCal;vAction)
  if(vAction=3) `did the user select an event or banner?
    CS_GetEvtSelect(eCal;vType;vIndex)
    Case of
      :(vType=1) `event
        ALERT("Event "+aEvent{vIndex}+" was selected.")
      :(vType=2) `banner
        ALERT("Banner "+aBanner{vIndex}+" was selected.")
    End Case
  End if `vAction=3)
End if `During
  
```

CS_GetSellItems

CS_GetSellItems(*area name; ItemType; SelectedItems*)

Parameter	Type	Description
<i>area name</i>	longint	name of CalendarSet object on layout
<i>ItemType</i>	integer	kind of item to return
<i>SelectedItems</i>	Integer array	index of all items belonging to selected days

CS_GetSellItems will return each of the array items that coincide with the selected items in the calendar. You will usually use this command after first using **CS_GetAction** ([page 53](#)) to determine that the user has made a selection.

ItemType — Integer. This parameter specifies the type of item to return information about.

- 1 events
- 2 icons

SelectedItems — Integer array. This parameter contains the index of each item in the array that corresponds to a selected item in the calendar. [See "FS_ArrayIntrscf" on page 86](#) for an example of this command.

Example:

```

C_LONGINT(vAction;vType;$i)
ARRAY INTEGER(aIndex;0)
if(During)
  
```

```

CS_GetAction(eCal;vAction)
If(vAction=0) `did the user select a day or days?
`get the events on this day or days
CS_GetSellItems(eCal;1;aIndex)
For($i;1;Size of array(aIndex))
`do something with each event's arrays
`for this example we'll put a bullet character at the beginning of each events
text
aEvent{aIndex{$i}}:="• "+aEvent{aIndex{$i}}
End for
End if `vAction=0)
End if `During

```

CS_LastClick

CS_LastClick(*area name*; *DateClicked*; *WasDouble*; *Modifiers*)

Parameter	Type	Description
<i>area name</i>	longint	name of CalendarSet object on layout
<i>DateClicked</i>	date	date that was clicked by user
<i>WasDouble</i>	integer	was event a double-click?
<i>Modifiers</i>	integer	modifier keys

CS_LastClick will return information on where the last click occurred.

DateClicked — Date. This parameter returns the date that was selected by the user. This value will be !00/00/00! if a banner is selected.

WasDouble — Integer. This parameter indicates if the user has double-clicked.

- 0 single-click
- 1 double-click

Modifiers — Integer. This parameter returns the values of any modifier keys which were pressed by the user when they clicked.

Value	Modifier Key
0	none
256	command
512	shift
1024	caps lock
2048	option
4096	control

Example:

```

`eCalendar script, detect a double-click on a day. If no modifier keys
`depressed, add an event. If command key depressed, modify an event
Case of
:(During)
  CS_LastClick (eCalendar;vDate;vDC;vMod)
  If(vDC=1) `was it a double-click?
    Case of
      :((vMod\256)%2=1) `command key?
        MOD CAL EVT (vDate)
    Else
      ADD CAL EVT (vDate)
    End case
  End if
End case

```

CS_GetResizBanr

CS_GetResizBanr(AreaName; BannerIndex)

Parameter	Type	Description
AreaName	longint	name of CalendarSet object on layout
BannerIndex	integer	array element corresponding to resized banner

CS_GetResizBanr is used to determine which banner was resized. You should call this command from the source area's script (or call-back procedure for an plug-in window) after first using **CS_GetAction** ([page 53](#)) and retrieving an action code of 5, indicating the user has resized a banner.

BannerIndex — Integer. Index of the resized banner within the arrays passed to CalendarSet using **CS_SetBanrArray** ([page 38](#)).

Note: CalendarSet automatically updates the start and end date array values to reflect the modified banner duration.

Example:

```

`eCal script
C_INTEGER(vActionCode;vEvtIndex)
CS_GetAction(eCal ;vActionCode) ` Determine user action
Case of
  :(vActionCode=5) ` User resized a banner
    CS_GetResizBanr(vCal;vEvtIndex)
  End case

```

CS_SetCallback

CS_SetCallback(*area name*; *ProcToExecute*)

Parameter	Type	Description
<i>area name</i>	longint	name of CalendarSet object on layout
<i>ProcToExecute</i>	string	procedure to execute when calendar is clicked

CS_SetCallback allows you to associate a procedure with a calendar area. The procedure will be executed whenever the calendar is clicked on. This command is normally used to detect and act on a user click on a CalendarSet plug-in window.

ProcToExecute — String. When CalendarSet detects a click on a calendar area, the procedure specified by *ProcToExecute* will be executed. CalendarSet will pass two parameters to the procedure.

The first parameter, *AreaName*, is passed to the callback procedure in \$1, and is of type long integer. *AreaName* contains the reference ID value of the CalendarSet object. This value, which is determined at runtime by 4D, is used for the first parameter of many of the CalendarSet commands, and is also the name of a CalendarSet plug-in object on a layout.

The second parameter, passed in \$2, represents the *ActionCode*, and is of type integer. The possible values for *ActionCode*, which are the same as the values returned by **CS_GetAction** ([page 53](#)), are shown below.

- 1 user click on day (no event selected),
- 2 user closed the window (valid only for CalendarSet plug-in windows)
- 3 user clicked on an event or banner
- 4 user dragged an event or banner
- 5 user resized a banner

When testing the value of the second parameter, a **Case** statement is recommended, to support additional action types which may be supported in a future version of CalendarSet.

Example:

```

`open a CalendarSet object in an plug-in window.
`configure the calendar for single day selection.
`highlight the entireday, show info in unused days, display month prefix.
`show the Current date in Times 14 point bold
`set the click procedure to be "HandleCalClick"
vCalWindow:=plug-in WINDOW(50; 50; 400; 400; 4; "Calendar Window";
    "_CS_Area") `open the window with CalendarSet displayed
    
```

```
CS_Options (vCalWindow;1;0;2;1)
CS_SetDayStyle (vCalWindow;Current date;"Times";14;1)
CS_SetCallback(vCalWindow;"HandleCalClick")
`
`GLOBAL PROCEDURE HandleCalClick
C_LONGINT($1;$CalArea) ` $1 = id of CalendarSet object
C_INTEGER($2;$Action) ` $2 = type of user action
$CalArea:=$1 `reassign the passed parameters for better readability
$Action:=$2
Case of
:($Action=1) `user click on a day
  CS_GetSelect ($CalArea;vDateSt;vDateEn;vContig;aDays)
  ModDayInfo ($CalArea;vDateSt) `add/modify the info for that day
:($Action=2) `user click in plug-in window close box
  SaveCalendar `save any changes
:($Action=3) `clicked on an event or banner

:($Action=4) `dragged an event or banner

:($Action=5) `resized a banner
End case
```

CS_DoResize

CS_DoResize (AreaName)

Parameter	Type	Description
<i>Area Name</i>	longin	name of AreaList object on layout

Use **CS_DoResize** only after having received an action code of 6, user resized the window, from **CS_GetAction** ([page 53](#)).

[See "Resizable Calendar Areas on a Form" on pag e31](#), and also ["Responding to User Actions on a CalendarSet Object" on page52](#) for more information.

CalendarSet Dragging Commands

Background

Technical Details of the Dragging Implementation

Currently, dragging is only allowed between plug-in areas. The user cannot drag to or from the Finder or other applications. Future versions of the *4D plug-in Drag Interface Specification* may define dragging with other applications.

You must configure CalendarSet to allow dragging out of and into a CalendarSet area. Commands provide the control necessary to allow dragging within an area, between two or more areas, and to not allow dragging between certain areas.

You can allow dropping onto days and events (and banners). For drags that originate from other areas, the dropping onto days could be conceived as the creation of a new event. Dropping onto events could be thought of as “linking” of some outside information to the event (such as linking a contact to an event as is done in many PIMs).

To allow dragging *out of* CalendarSet, you must pass an access “code” for the type of data that is to be dragged. You must specify the type of data to allow to be dragged and at least one code to enable dragging, using ***CS_SetDrgSrc*** (page 64). Currently, only events can be dragged out of CalendarSet, so only the event data type is allowed. Up to ten codes can be passed. Allowing many codes provides for more flexibility in enabling and disabling dragging between various areas. This will be explained in more depth later.

In order to allow dragging *into* CalendarSet, you must pass an access “code” for the type of data that can be the destination of a drag. You must specify the type of data that can receive a drag, and at least one code to enable dragging, using ***CS_SetDrgDst*** (page 65). CalendarSet supports dragging to both days and events. As with ***CS_SetDrgSrc***, up to ten codes can be passed for flexibility reasons.

What are access “codes”?

The access codes that are passed in ***CS_SetDrgSrc*** (page 64) and ***CS_SetDrgDst*** (page 65) are used to enable dragging between specific drag partners. These drag partners can be the same CalendarSet area, different CalendarSet areas, or different plug-in areas.

When a drag takes place, the drag sender’s plug-in code communicates its access codes to the drag receiver’s plug-in code. The drag receiver will compare the access codes of the sender to its own codes. If any of the codes match, the drag is allowed. This mechanism allows a number of combinations between several drag partners. The following is an example of enabling the dragging of events within the same CalendarSet area.

```
`enable drag events to days within the this area  
vSelfStr:=String(eCal) ` creates a unique code that only allows dragging within  
this area
```

```
CS_SetDrgSrc(eCal;2;vSelfStr) ` event data type for source
```

```
CS_SetDrgDst(eCal;1;vSelfStr) ` day data type for destination
```

This example also shows how you can create a unique identifier that only enables dragging within the same CalendarSet area.

In the case when the sender and the receiver is the same CalendarSet area and the source of the drag is an event and the destination is a day, then CalendarSet will automatically update the event date array and the banner start and end date arrays. For banners, the banner’s day range will be moved to the a new range of days by determining a difference in days from where it was originally

selected and where it was dragged to (i.e., if the banner is selected on the last day of its duration and moved to a new day, the destination of the drag will become the last day of the banner's new location).

After the drag

When an event is dragged out of CalendarSet, the following information is available to you:

- ◆ Notification that a drag occurred
- ◆ Which event was dragged (index in array) and what type (regular, banner) was dragged
- ◆ Where the event was dragged to (this area or another area)

When the drag is completed, the CalendarSet script (or callback procedure) is executed. You can then determine what the last user action was using ***CS_GetAction*** ([page 53](#)). If a drag occurred, you can determine which event was dragged using ***CS_GetDrgSrcEvt*** ([page 67](#)). This command indicates which event was dragged and its type (event or banner).

To determine which plug-in area was the destination of the drag, call ***CS_GetDrgArea*** ([page 68](#)). This command returns the *AreaName* (a long integer) and the process id of the destination area, which may be the same CalendarSet area, another CalendarSet area, or another plug-in area. When dragging to another object, that object can either reside in the same window or on another window, which may require use of 4D's **CALL PROCESS** and **Outside call** commands to take action on the drag — *when dragging to other objects, CalendarSet is only providing a user interface to the drag, and notifying you, the developer, that the drag has occurred. You are responsible for manipulating any arrays or other data structures.*

When a CalendarSet area is the destination of a drag, the following information is available to you:

- ◆ The type of data that was the recipient of the drag (either day or event)
- ◆ If the type is a day, the date of that day
- ◆ If the type is an event, the type of event (event or banner) and the event index within its respective arrays

You must call ***CS_GetDrgDstTyp*** ([page 69](#)) to determine if the destination of the drag was either a day or event. If the destination was a day, ***CS_GetDrgDstDay*** ([page 70](#)) is used to determine the destination date. If the destination was an event, ***CS_GetDrgDstEvt***

[\(page 71\)](#) is called to determine which event was the destination.

If the destination of the drag is an area on another window, then you must use 4D's **CALL PROCESS** and **Outside call** commands to communicate to the other process.

NOTE: Keep in mind that CalendarSet will update the arrays and refresh the area if the drag is within the same area (events or banners to days).

CalendarSet on Multi-page Layouts

You can place a CalendarSet area on layouts that contain multiple pages. If you've configured the area to accept a drag from another area, you must enable and disable the CalendarSet area using **Area_SetEnable** [\(page 88\)](#), depending on whether the area is on the current page. If the page containing the CalendarSet area is not the current page, call **Area_SetEnable** with a parameter of 0 to disable dragging onto the CalendarSet area. When the page becomes current, call **Area_SetEnable** with a parameter of 1 to re-enable the ability to accept a drag.

Note: You should always disable a CalendarSet area which is not on the current layout page.

CalendarSet DataType

This DataType represents the type of the drag for both the source and the destination. It is used in the commands **CS_SetDrgSrc** [\(page 64\)](#), **CS_SetDrgDst** [\(page 65\)](#), and **CS_GetDrgDstTyp** [\(page 69\)](#). These are the possible values:

- 0 No drag
- 1 Day
- 2 Event

CS_DragMgrAvail

CS_DragMgrAvail(IsDragMgrPresent)

Parameter	Type	Description
<i>IsDragMgrPresent</i>	longint	indicates whether Drag Manager is installed

CS_DragMgrAvail is used to determine if the Drag Manager is installed, and alert the user or take some other action.

IsDragMgrPresent — Integer, 0 or 1.

- 1** the Drag Manager is present on this machine, and CalendarSet dragging and banner resizing features are available
- 0** the Drag Manager is *not* present on this machine, and CalendarSet dragging and banner resizing features cannot be used

This command can be called from any 4D method, even if a CalendarSet object is not displayed.

Example:

```
C_LONGINT(vDragMgr)
CS_DragMgrAvail(vDragMgr)
If(vDragMgr=0)
  BEEP
  ALERT("The Macintosh Drag Manager is not installed, you will be unable to
  drag events or banners!")
End if
```

CS_SetDrgSrc

CS_SetDrgSrc(AreaName; SourceDataType; SrcCode1; SrcCode2; ... ; SrcCode10)

Parameter	Type	Description
<i>AreaName</i>	longint	name of CalendarSet object on layout
<i>SourceDataType</i>	integer	type of item dragged
<i>SrcCodeN</i>	string	used to match drag partners

CS_SetDrgSrc is used to enable dragging out of the CalendarSet object *AreaName*, by setting the access codes for the source of the drag. This command *must* be called before a drag is initiated (usually in the **Before** phase). *Please read the section "What are access codes?" on page 61 for more information.*

SourceDataType — Integer, 2. Currently, only events can be dragged from CalendarSet. Default is 2. Future versions of CalendarSet may allow other types of dragging. Possible values are:

- 0** No drag
- 1** Day
- 2** Event

SrcCode — String (15 characters). The *SrcCode* can have any

value, such as “MonthDrag”, “WeekDrag”, “DragToALP”, etc.; however, it is meant to match a code passed into a potential drag partner. The drag partner will be the destination/receiver of the drag. That destination can be the same CalendarSet area, a different CalendarSet area, or another plug-in object.

This code can be any value other than an empty string. Avoid using the strings “TEXT” or “PICT”.

CalendarSet performs the following logic during the actual drag. When the drag takes place, the source codes that were given in *SrcCode1*, *SrcCode2*, etc. will be communicated to the receiver of the drag. If any of the codes match, the drag is enabled.

[See “What are access “codes”?” on page 61.](#)

Examples:

```

`enable dragging events to days within the this area
vSelfStr:=String(eCal) ` creates a unique code that only allows dragging within
this area
CS_SetDrgSrc (eCal;2;vSelfStr) ` event data type for source
CS_SetDrgDst (eCal;1;vSelfStr) ` day data type for destination

`enable dragging from a week view into another CalendarSet area (month view)
CS_SetDrgSrc (eWeekCal;2;"WeekToMonth") ` event data type for source

```

CS_SetDrgDst

CS_SetDrgDst(*AreaName*; *DestDataType*; *DstCode1*; *DstCode2*; ... ; *DstCode10*)

Parameter	Type	Description
<i>AreaName</i>	longint	name of CalendarSet object on layout
<i>DestDataType</i>	integer	data type to be received
<i>DstCodeN</i>	string	access code(s) to be received

CS_SetDrgDst is used to enable dragging into the destination area, by setting the access codes. *Please read the section “What are access “codes”?” on page 61 for more information.*

This command must be called *before* a drag has occurred.

The *AreaName* parameter must be the destination (receiver) area of a drag.

DestDataType - Integer, 1 or 2.

- 1 days
- 2 events

For the data type specified by *DestDataType* (either day or event), you must specify at least one *DstCode* to enable receiving of that type.

DstCode - String (15 characters). The *DstCode* can be any value (other than an empty string), such as “MonthDrag”, “WeekDrag”, “ALPDrag”, etc. Avoid using the strings “TEXT” or “PICT”.

The code should be the same as what is passed into a potential drag partner. The drag partner will be the source/sender of the drag. The source area can be the same CalendarSet area, a different CalendarSet area, or another plug-in object.

CalendarSet performs the following logic during the actual drag. When the drag takes place, the destination codes that were given in *DstCode1*, *DstCode2*, etc. are compared to the source codes communicated by the sender of the drag. If any of the codes match, the drag is enabled.

[See “Technical Details of the Dragging Implementation” on page 60.](#)

*Note: When CalendarSet is placed on a page in a multi-page layout, be sure to disable dragging by calling **Area_SetEnable** ([page 88](#)) when that page is not the currently shown page. Please read the section “After the drag” on page e62 for more information.*

Example:

```
` enable dragging events to days within this area
vSelfStr:=String(eCal) ` creates a unique code that only allows dragging within
                        this area
CS_SetDrgSrc(eCal;2;vSelfStr) ` event data type for source
CS_SetDrgDst(eCal;1;vSelfStr) ` day data type for destination

` enable dragging into a month view from another CalendarSet area (week view )
CS_SetDrgDst(eMonthCal;1;"WeekToMonth") ` allow dragging into days
CS_SetDrgDst(eMonthCal;2;"WeekToMonth") ` allow dragging into events and
                        banners
```

CS_GetDrgSrcEvt

CS_GetDrgSrcEvt(AreaName; EventType; EventIndex; StartDate)

Parameter	Type	Description
<i>AreaName</i>	longint	name of CalendarSet object on layout
<i>EventType</i>	integer	type of item dragged
<i>EventIndex</i>	integer	index into corresponding array element
<i>StartDate</i>	date	starting date of the drag

Use ***CS_GetDrgSrcEvt*** to determine which event was dragged after a drag has completed. The *AreaName* parameter should be the source (sender) area of a drag. This command is called from the source area's script (or callback procedure for plug-in windows) after receiving an action code of 4 — user dragged event.

EventType — Integer, 1 or 2. Values are shown below.

- 1 event
- 2 banner

EventIndex — Integer. This parameter is an index within the array that was passed to CalendarSet via either ***CS_SetArray*** ([page 35](#)) or ***CS_SetBanrArray*** ([page 38](#)).

StartDate — Date. This parameter is the starting date of the drag. For events, this date will be equal to the event's date passed in ***CS_SetArray*** ([page 35](#)). For banners, this date will be the starting point of the drag, which will be some value between the banner's start and end dates, inclusively.

Examples:

```
`eSrcCal script
C_LONGINT(vDstArea)
C_INTEGER(vActionCode;vDstID;vEvtType;vEvtIndex)
C_DATE(vDate)

CS_GetAction(eSrcCal;vActionCode) ` Determine user action
Case of
:(vActionCode=4)pp` User dragged an event
  CS_GetDrgSrcEvt (eSrcCal;vEvtType;vEvtIndex;vDate)
End case
```

CS_GetDrgArea

CS_GetDrgArea(AreaName; DestArea; DestProcessID)

Parameter	Type	Description
<i>AreaName</i>	longint	name of CalendarSet object on layout
<i>DestArea</i>	integer	ID of the area the item was dragged to
<i>DestProcessID</i>	integer	process ID of the <i>DestArea</i>

Use **CS_GetDrgArea** to determine the destination area of the last drag. The *AreaName* parameter should be the source (sender) area of a drag. This command is called from the source area's script (or callback procedure for plug-in windows) after receiving a action code of 4 — user dragged event.

DestArea — Integer. This parameter is the area reference of the area that is the destination of the drag.

DestProcessID — Integer. This parameter contains the Process ID in which the window and destination area reside. Use the 4D commands **CALL PROCESS** and **Outside call** for interprocess communication.

Note: If the DestProcessID is different from the current process, you will need to use the 4D CALL PROCESS and Outside call commands to communicate to the window that contains the destination area.

Example:

```
`eSrcCal script
C_LONGINT(vDstArea)
C_INTEGER(vActionCode;vDstID;vEvtType;vEvtIndex)
C_DATE(vDate)
```

CS_GetAction(eSrcCal ;vActionCode) ` Determine user action

Case of

:(vActionCode=4)pp` User dragged an event

CS_GetDrgSrcEvt(eSrcCal ;vEvtType;vEvtIndex;vDate)

CS_GetDrgArea(eSrcCal;vDstArea;vDstID)

If (vDstID#**Current process**) ` if dragged to a different process

 bvDstArea:=vDstArea ` store in interprocess variable

CALL PROCESS(vDstID)

End if

End case

CS_GetDrgDstTyp

CS_GetDrgDstTyp(AreaName;DestDataType)

Parameter	Type	Description
<i>AreaName</i>	longint	name of CalendarSet object on layout
<i>DestDataType</i>	integer	type of data which was destination of drag

CS_GetDrgDstTyp is used to determine the type of data that was the destination of the last drag. Specifically, the user may drag items to either a day's cell, an event, or a banner. After the drag has completed, ***CS_GetDrgDstTyp*** indicates whether the destination of the drag was a day, event, or banner. The *AreaName* parameter should be the destination (receiver) area of a drag.

If the destination and source areas are actually the same area or different areas within the same process (i.e., they reside on the same layout), this command may be called from the source area's script (or callback procedure for plug-in windows). If the destination and source areas are in different processes, then you will need to use the 4D **CALL PROCESS** and **Outside call** commands and interprocess variables to communicate between the two processes.

DestDataType — Integer, 1 or 2. Indicates what type of data was the destination of the last drag. Values are shown below.

- 1 day
- 2 event or banner

Example:

```

`eSrcCal script
C_LONGINT(vDstArea)
C_INTEGER(vActionCode;vDstID;vEvtType;vEvtIndex;vDstType)
C_DATE(vDate)

CS_GetAction(eSrcCal;vActionCode) ` Determine user action
Case of
:(vActionCode=4)pp ` User dragged an event
  CS_GetDrgSrcEvt(eSrcCal;vEvtType;vEvtIndex;vDate)
  CS_GetDrgArea(eSrcCal;vDstArea;vDstID)
  If(vDstID=Current process) ` was the drag to an object in this process?
    If (vDstArea=eSrcCal ) ` if dragged within the same area
      CS_GetDrgDstTyp(eSrcCal;vDstType) ` get the type of data that was destination of the drag
      If(vDstTyp=1) ` if dragged into a day
        CS_GetDrgDstDay(eSrcCal;vDate) ` get day's date
      End if
    End if `vDstID=Current process
  Else `dragged to a different process

```

```

        pVDstArea:=vDstArea
        CALL PROCESS(vDstID)
    End if
End case

```

```

    `Destination calendar layout's layout proc
    C_INTEGER(vEvtType;vEvtIndex;vDstType)
    C_DATE(vDate)

```

Case of

```

:(Outside call)pp ` Outside call (via CALL PROCESS)
If(pVDstArea=eDstCal) ` has a drag occurred from another process into this
    calendar
    CS_GetDrgDstTyp(eDstCal;vDstType) ` get the type of data that was desti-
        nation of the drag
    If(vDstTyp=1) ` if dragged into a day
        CS_GetDrgDstDay(eDstCal;vDate) ` get day's date
    Else
        CS_GetDrgDstEvt(eDstCal;vEvtType;vEvtIndex) ` get drop event
    End if
End if
End case

```

CS_GetDrgDstDay

CS_GetDrgDstDay(AreaName; DestDate)

Parameter	Type	Description
<i>AreaName</i>	longint	name of CalendarSet object on layout
<i>DestDate</i>	integer	day which received the last drag

If the destination of the last drag was a day ([See “CS_GetDrgDstTyp” on page 69](#)), use *CS_GetDrgDstDay* to determine the destination date of the last drag. The *AreaName* parameter should be the destination (receiver) area of a drag.

If the destination and source areas are actually the same area or different areas that within the same process (i.e., they reside on the same layout), this command may called from the source area's script (or callback procedure for plug-in windows). If the destination and source areas are in different processes, then you will need to use the 4D **CALL PROCESS** and **Outside call** commands and interprocess variables to communicate between the two processes.

DestDate — Date. The day that received the last drag.

Example:

```

    `eSrcCal script
    C_LONGINT(vDstArea)

```

```

C_INTEGER(vActionCode;vDstID;vEvtType;vEvtIndex;vDstType)
C_DATE(vDate)

CS_GetAction(eSrcCal;vActionCode) ` Determine user action
Case of
:(vActionCode=4)pp` User dragged an event
  CS_GetDrgSrcEvt(eSrcCal;vEvtType;vEvtIndex;vDate)
  CS_GetDrgArea(eSrcCal;vDstArea;vDstID)
  If(vDstID=Current process) ` was the drag to an object in this process?
    If (vDstArea=eSrcCal ) ` if dragged within the same area
      CS_GetDrgDstTyp(eSrcCal;vDstType) ` get the type of data that was des-
        tination of the drag
      If(vDstTyp=1) ` if dragged into a day
        CS_GetDrgDstDay(eSrcCal;vDate) ` get day's date
      End if
    End if `vDstID=Current process
  Else `dragged to a different process
    pvDstArea:=vDstArea
    CALL PROCESS(vDstID)
  End if
End case

` Destination calendar layout's layout proc
C_INTEGER(vEvtType;vEvtIndex;vDstType)
C_DATE(vDate)

Case of
:(Outside call)pp` Outside call (via CALL PROCESS)
  If(pvDstArea=eDstCal) ` has a drag occurred from another process into this
    calendar
    CS_GetDrgDstTyp(eDstCal;vDstType) ` get the type of data that was desti-
      nation of the drag
  If(vDstTyp=1) ` if dragged into a day
    CS_GetDrgDstDay(eDstCal;vDate) ` get day's date
  Else
    CS_GetDrgDstEvt(eDstCal;vEvtType;vEvtIndex) ` get drop event
  End if
End if
End case

```

CS_GetDrgDstEvt

CS_GetDrgDstEvt(AreaName; EventType; EventIndex)

Parameter	Type	Description
AreaName	longint	name of CalendarSet object on layout
EventType	integer	event or banner
EventIndex	integer	index into corresponding array element

If the destination of the last drag was an event ([See "CS_GetDrgDstTyp" on page 69](#)), use this command to determine

which event was the destination of the last drag. The *AreaName* parameter should be the destination (receiver) area of a drag.

If the destination and source areas are actually the same area or different areas that within the same process (i.e., they reside on the same layout), this command may be called from the source area's script (or callback procedure for plug-in windows). If the destination and source areas are in different processes, then you will need to use the 4D **CALL PROCESS** and **Outside call** commands and interprocess variables to communicate between the two processes.

EventType — Integer, 1 or 2. Values are shown below.

- 1 event
- 2 banner

EventIndex — Integer. Index into the arrays that were passed to CalendarSet via either **CS_SetArray** ([page 35](#)) or **CS_SetBannerArray** ([page 38](#)).

Example:

```

`eSrcCal script
C_LONGINT(vDstArea)
C_INTEGER(vActionCode;vDstID;vEvtType;vEvtIndex;vDstType)
C_DATE(vDate)

CS_GetAction(eSrcCal ;vActionCode) ` Determine user action
Case of
:(vActionCode=4)pp ` User dragged an event
  CS_GetDrgSrcEvt(eSrcCal ;vEvtType;vEvtIndex;vDate)
  CS_GetDrgArea(eSrcCal ;vDstArea;vDstID)
  If(vDstID=Current process) ` was the drag to an object in this process?
    If (vDstArea=eSrcCal ) ` if dragged within the same area
      CS_GetDrgDstTyp(eSrcCal;vDstType) ` get the type of data that was destination of the drag
      If(vDstTyp=1) ` if dragged into a day
        CS_GetDrgDstDay(eSrcCal;vDate) ` get day's date
      End if
    End if `vDstID=Current process
  Else `dragged to a different process
    pvDstArea:=vDstArea
    CALL PROCESS(vDstID)
  End if
End case

`Destination calendar layout proc
C_INTEGER(vEvtType;vEvtIndex;vDstType)
C_DATE(vDate)

Case of
:(Outside call)pp ` Outside call (via CALL PROCESS)

```

CalendarSet Dragging Commands

```
If(≠vDstArea=eDstCal) ` has a drag occurred from another process into this
calendar
  CS_GetDrgDstTyp(eDstCal;vDstType) ` get the type of data that was desti-
  nation of the drag
  If(vDstTyp=1) ` if dragged into a day
    CS_GetDrgDstDay(eDstCal;vDate) ` get day's date
  Else
    CS_GetDrgDstEvt(eDstCal;vEvtType;vEvtIndex) ` get drop event
  End if
End if
End case
```

Popup Commands

CalendarSet provides several popups to assist in implementing an intuitive user interface built around the calendar object.

Using the Date and Time Selection Popups

Date and time selection popups are provided to enhance the user interface of entry and modification of date and time values. With both popups, two commands are available to set the current value, as well as to get the value selected by the user. [See “Date and Time Selection Popups” on page 15.](#)

The date popup command to set the current value is ***MM_SetDate*** ([page 76](#)), and the command to get the value selected by the user is ***MM_GetDate*** ([page 77](#)).

The time popup command to set the current value is ***TM_SetTime*** ([page 78](#)), and the command to get the value selected by the user is ***TM_GetTime*** ([page 79](#)).

Using the Color Selection Popup

A color selection popup is available to aid the development of an interface for selecting color of calendar data. [See “Color Selection Popup” on page 16.](#)

When running in 256 color mode, the popup displays a palette of eight standard colors as well as 4D’s 256 color palette. The palette will configure itself automatically to display fewer colors if the Mac is running in less than a 256 color mode.

Table 7:

Color	Value
Black	-1
White	-2
Red	-3
Green	-4
Blue	-5
Cyan	-6
Magenta	-7

Table 7:

Color	Value
Yellow	-8
4D Colors	1 to 256

The 4D color palette is a 16 by 16 grid. To determine a color's value, you can locate the color's position on the color grid in the Design environment (the Color submenu which is available in the Layout and Procedure editors), and count the number of rows down and columns across. The equation is

$$\text{ColorValue} = ((\text{RowNumber} - 1) \times 16) + \text{ColumnNumber}$$

The color popup includes the **CM_SetColor** command ([page 80](#)) for setting the current color, and the **CM_GetColor** command ([page 81](#)) to get the user selected color.

Using the Icon Selection Popup

An icon selection popup is included to display icon resources in a palette. This provides an elegant interface tool for allowing user selection of icon data for a calendar. [See "Icon Selection Popup" on page 17.](#)

The icon popup includes the **IM_SetArray** command ([page 82](#)) for specifying what icons to include in the popup and the size of the icons, the **IM_SetSelect** command ([page 83](#)) for setting the current popup item, and the **IM_GetSelect** command ([page 82](#)) for getting the user selected item.

Commands

%MM_PopArea

%MM_PopArea allows you to implement a calendar popup.

MM_Options

MM_Options(AreaName; HidePopArrow)

Parameter	Type	Description
AreaName	longint	name of the %MM_PopArea object on layout
HidePopArrow	integer	hide the popup triangle indicator

MM_Options is used to configure the calendar popup object specified by *AreaName*.

HidePopArrow — Integer, 0 or 1. This parameter lets you hide the triangle symbol which is displayed when the popup is not be clicked. You may desire a different appearance of the pop-up, such as placing a picture or icon below the plug-in object.

- 1 hide the triangle symbol that is drawn on the layout that indicates that the area is a pop-up
- 0 show the symbol (default)

Examples:

```
`hide the triangle symbol
MM_Options(eDatePop;1)
`show the triangle symbol
MM_Options(eDatePop;0)
```

MM_SetDate

MM_SetDate(area name; DateToSet)

Parameter	Type	Description
area name	longint	name of the MM_PopArea object on layout
DateToSet	date	date the calendar popup should display when clicked

MM_SetDate allows you to specify which date the popup should have selected when first clicked (popped up).

DateToSel — Date. This parameter is used to specify the current selected date when the popup is clicked.

Example:

```
`date popup script ("only if modified" checkbox is turned off)
if(Before)
if(Record number([Invoice])=-3) `is this a new record?
    MM_SetDate(ePopDate;Current date)
    [Invoice]Invoice Date:=Current date
```

```

Else
  MM_SetDate(ePopDate:[Invoice]Invoice Date)
End if
End if

```

MM_GetDate

MM_GetDate(area name; DateToGet)

Parameter	Type	Description
<i>area name</i>	longint	name of the %MM_PopArea object on layout
<i>DateToGet</i>	date	currently selected date in the calendar popup

MM_GetDate allows you to retrieve the date that the user selected the last time area name was popped up.

DateToGet — Date. This parameter is the date selected by the user.

Example:

```

`date popup script
If(During)
  MM_GetDate(ePopDate;vDate)
  [Invoice]Invoice Date:=vDate
End if

```

%TM_PopArea

%TM_PopArea allows you to implement a Time popup menu. This popup allows the user to select a time in 5 minute increments.

*Note: The variables that are passed to **TM_SetTime** ([page 78](#)) and **TM_GetTime** ([page 79](#)) are not time variables, but rather string variables that are the names of the time variables. This is done due to a bug in 4th Dimension 2.x and versions of 4th Dimension 3.0 thru 3.0.2.*

TM_Options

TM_Options(AreaName; HidePopArrow)

Parameter	Type	Description
<i>AreaName</i>	longint	name of the %TM_PopArea object on layout
<i>HidePopArrow</i>	integer	hide the popup triangle indicator

TM_Options is used to configure the calendar popup object specified by *AreaName*.

HidePopArrow — Integer, 0 or 1. This parameter lets you hide the triangle symbol which is displayed when the popup is not be clicked. You may desire a different appearance of the pop-up, such as placing a picture or icon below the plug-in object.

- 1 hide the triangle symbol that is drawn on the layout that indicates that the area is a pop-up
- 0 show the symbol (default)

Examples:

```
`hide the triangle symbol
TM_Options(eTimePop;1)
`show the triangle symbol
TM_Options(eTimePop;0)
```

TM_SetTime

TM_SetTime(*area name*; *TimeToSet*)

Parameter	Type	Description
<i>area name</i>	longint	name of time popup object on layout
<i>TimeToSet</i>	string	time to preselect when popup clicked

TM_SetTime allows you to specify the time the popup should have selected when first popped up. The time will default to the current time if this command is not issued.

TimeToSet — String. This parameter is the *name* of a time variable to use to set the current time of the popup.

Example:

```
`initialize the popup eCallTime to the value in the Call Time field
If([Call History]Call Time#†00:00:00†) `if not a new record
  vTime:=[Call History]Call Time
Else
  vTime:=Current time
End if
TM_SetTime(eCallTime;"vTime")
```

TM_GetTime

TM_GetTime(*area name*; *TimeToGet*)

Parameter	Type	Description
<i>area name</i>	longint	name of time popup object on layout
<i>TimeToGet</i>	string	time selected last time popup clicked

TM_GetTime allows you to retrieve the time that the user selected the last time the popup was clicked.

TimeToGet — String. This parameter is the *name* of a time variable, which contains the time selected by the user.

Example:

```
`get the time selected in the eCallTime popup
TM_GetTime (eCallTime;"vCallTime")
[Call History]Call Time:=vCallTime
```

%CM_PopArea

%CM_PopArea is an plug-in area which allows you to implement a color palette popup on a 4D layout. This popup allows the user to select a color from the 8 colors listed below, or any color in the 4D color palette.

Table 8:

Color	Value
Black	-1
White	-2
Red	-3
Green	-4
Blue	-5
Cyan	-6
Magenta	-7
Yellow	-8
4D Colors	1 to 256

If the monitor that is being used is 8 bits in depth or higher, the popup will display 256 colors. If the monitor is 4 bits, the palette will display 16 colors. If it is 2 bits or lower the palette will be replaced by

a regular menu that gives the user a choice of the 8 primary colors.

CM_SetColor

CM_SetColor(*area name*; *ColorToSet*)

Parameter	Type	Description
<i>area name</i>	longint	name of color popup object on layout
<i>ColorToSet</i>	integer	desired color

CM_SetColor allows you to specify which color the popup should have selected when first popped up. The default color is black.

ColorToSet — Integer. This parameter is used to set the currently selected color for the popup.

Table 9:

Color	Value
Black	-1
White	-2
Red	-3
Green	-4
Blue	-5
Cyan	-6
Magenta	-7
Yellow	-8
4D Colors	1 to 256

The 4D color palette is a 16 by 16 grid. To determine a color's value, you can locate the color's position on the color grid in the Design environment (the Color submenu which is available in the Layout and Procedure editors), and count the number of rows down and columns across. The equation is

$$\text{ColorValue} = ((\text{RowNumber} - 1) \times 16) + \text{ColumnNumber}$$

Example:

```
`set the color for the eColor popup to the value in [Calendar]Color
CM_SetColor(eColor;[Calendar]Color)
`set the color for the eColor popup to Red
CM_SetColor(eColor;-3)
```

CM_GetColor

CM_GetColor(area name; ColorToGet)

Parameter	Type	Description
<i>area name</i>	longint	name of color popup object on layout
<i>ColorToGet</i>	integer	currently selected color in the popup object

CM_GetColor allows you to retrieve the color that the user selected the last time area name was popped up.

ColorToGet — Integer. Use this parameter to identify what color the user selected.

Table 10:

Color	Value
Black	-1
White	-2
Red	-3
Green	-4
Blue	-5
Cyan	-6
Magenta	-7
Yellow	-8
4D Colors	1 to 256

The 4D color palette is a 16 by 16 grid. To determine a color's value, you can locate the color's position on the color grid in the Design environment (the Color submenu which is available in the Layout and Procedure editors), and count the number of rows down and columns across. The equation is

$$\text{ColorValue} = ((\text{RowNumber} - 1) \times 16) + \text{ColumnNumber}$$

Example:

```
`get the currently selected color in the eColor popup
CM_GetColor(eColor;vColor)
[Calendar]Color:=vColor
```

%IM_PopArea

%IM_PopArea allows you to implement an icon popup menu.

The icons that are displayed by the popup are specified by an array of resource IDs. The icons that are used by **%IM_PopArea** are the 'icl8', 'icl4', 'ICN#', 'ics8', 'ics4', and 'ics#' resources that the Finder uses to display icons.

IM_SetArray

IM_SetArray(*area name*; *ArrayName*; *IconSize*; *NumColumns*)

Parameter	Type	Description
<i>area name</i>	longint	name of icon popup object on layout
<i>ArrayName</i>	string	name of the array of indices
<i>IconSize</i>	integer	size to use for the icon
<i>NumColumns</i>	integer	number of columns of icons to display in the popup

IM_SetArray specifies the array to use when displaying the popup of icons.

ArrayName — String (name of an integer array). This parameter is used to pass the resource id's of the icons that are to be displayed in the popup menu.

IconSize — Integer. This parameter is the size of the icon to use and can be either 32 (default) for large icons or 16 for small icons.

NumColumns — Integer. Specifies the number of columns to display when the popup is displayed. If *NumColumns* is 0 the popup will have an equal number of rows and columns.

Example:

```
`display the icons specified by the icon IDs in the array alcons
`the icon object is elcon, and the icons should be displayed as large icons
`the icon popup should have an equal number of rows and columns
IM_SetArray(elcon;"alcons";32;0)
```

IM_GetSelect

IM_GetSelect(*area name*; *CurrentSelectedItem*)

Parameter	Type	Description
<i>area name</i>	longint	name of icon popup object on layout
<i>CurrentSelectedItem</i>	integer	index of the selected icon

IM_GetSelect allows you to retrieve the icon that the user selected the last time it was popped up. Use *CurrentSelectedItem* as an index into the icon array to determine the resource id of the selected

icon.

CurrentSelectedItem — Integer. This parameter returns an index into the icon array, indicating the icon selected by the user.

Example:

```
`get the Icon ID for the currently selected icon in the elcon object  
IM_GetSelect(elcon;vIndex)  
[Calendar]Icon ID:=alcons{vIndex}
```

IM_SetSelect

IM_SetSelect(area name; CurrentSelectedItem)

Parameter	Type	Description
<i>area name</i>	longint	name of icon popup object on layout
<i>CurrentSelectedItem</i>	integer	index into icon array

IM_SetSelect allows you to set the currently selected item in the popup.

CurrentSelectedItem — Integer. This parameter is used to set the currently selected icon in the popup. Default is 1.

Example:

```
`set the icon popup elcon to display the icon saved in [Calendar]Icon ID  
$ID_Index:=Find in array(alcons;[Calendar]Icon ID)  
if($ID_Index>0) `make sure the icon id is in the array  
    IM_SetSelect(elcon;$ID_Index)  
End if
```

CalendarSet Utility Commands

CalendarSet includes several commands to assist in managing the operation of a calendar area, as well as implement other user interface objects.

International Utilities

Dates are handled differently in different countries. CalendarSet provides several commands to assist with dates. These commands use the Macintosh Toolbox date routines, and are aware of the location set in the Macintosh Date and Time Control Panel.

To obtain the correct day name for the currently selected location, use **CS_GetDayName** ([page 85](#)). Use **CS_GetMonthName** ([page 85](#)) to get the correct month name. Use **Util_SetDate** ([page 87](#)) to set the value of a date field or variable in an international-aware way.

Array Intersection

You can intersect two arrays to create a third array using **FS_ArrayIntrsc** ([page 86](#)).

CalendarSet on Multi-page Layouts

You can place a CalendarSet area on layouts that contain multiple pages. If you've configured the area to accept a drag from another area, you must enable and disable the CalendarSet area using **Area_SetEnable** ([page 88](#)) based on whether the area is on the current page. If the page containing the CalendarSet area is not the current page, call **Area_SetEnable** with a parameter of 0 to disable dragging onto the CalendarSet area. When the page becomes current, call **Area_SetEnable** with a parameter of 1 to re-enable the ability to accept a drag.

For more information about CalendarSet dragging, please see [“CalendarSet Dragging Commands” on page e60](#).

Note: for compatibility with future versions of CalendarSet you should always disable a CalendarSet area which is not on the current layout page.

CS_GetDayName

CS_GetDayName(DayNumber; Abbreviated; DayName)

Parameter	Type	Description
<i>DayNumber</i>	integer	day number to get the name of
<i>Abbreviated</i>	integer	get abbreviated day name
<i>DayName</i>	string	day name

CS_GetDayName returns in *DayName* the day of the week specified by *DayNumber*.

DayNumber — Integer. This is the number returned by the 4D function **Day number** and is in the range of 1 to 7 for the days Sunday thru Saturday.

Abbreviated — Integer. This parameter is used to specify whether you want the full day name, or an abbreviation. If *Abbreviated* is set to 1, *DayName* will only contain as many characters as is appropriate for abbreviations, according to the Macintosh System software that is being used. For example, Tuesday abbreviated in the United States is “Tue”.

DayName — String. This parameter returns the name of the day specified by *DayNumber*.

Examples:

```

`fill an array with the full names of all days
C_INTEGER($i)
C_STRING(20;vDayName)
ARRAY STRING(20;aDayNames;7)
For($i;1;7)
    CS_GetDayName ($i;0;vDayName)
    aDayNames{$i}:vDayName
End for

```

CS_GetMonthName

CS_GetMonthName(MonthNumber; Abbreviated; MonthName)

Parameter	Type	Description
<i>MonthNumber</i>	integer	month number to get the name of
<i>Abbreviated</i>	integer	get abbreviated month name
<i>MonthName</i>	string	month name

CS_GetMonthName returns in *MonthName* the name of the month

specified by *MonthNumber*.

MonthNumber — Integer. This parameter is a number between 1 and 12, corresponding to the months January thru December.

Abbreviated — Integer. This parameter is used to indicate whether to abbreviate the value returned in *MonthName*. If *Abbreviated* is set to 1, *MonthName* will only contain as many characters as is appropriate for abbreviations according to the Macintosh System software that is being used. For example, “February” is abbreviated in the United States as “Feb”.

MonthName — String. This parameter returns the name of the month specified by *MonthNumber*.

Examples:

```
`fill an array with the full names of all months
C_INTEGER($i)
C_STRING(20;vMonthName)
ARRAY STRING(20;aMonthName;12)
For($i;1;12)
    CS_GetMonthName ($i;0;vMonthName)
    aMonthNames{$i}:vMonthName
End for
```

FS_ArrayIntrsect

FS_ArrayIntrsect(*CompareArray*; *SourceArray*; *DestArray*)

Parameter	Type	Description
<i>CompareArray</i>	integer array	array of indices into source array
<i>SourceArray</i>	array	array to extract items from
<i>DestArray</i>	array	array to place intersected items into

FS_ArrayIntrsect allows you to build an array based upon the intersection of an Integer array of indices and any other array. This command is useful with the CalendarSet routine ***CS_GetSellItems*** ([page 55](#)). It allows you to quickly build a new array out of the items that correspond to the selected dates in the calendar.

CompareArray — Integer array. This parameter is an array of indices into the *SourceArray*.

SourceArray — Array (any type). The original array.

DestArray — Array (any type). The results of the intersection are placed into this array.

Example:

```

C_LONGINT(vAction;vType;$i)
ARRAY INTEGER(aIndex;0)
If(During)
  CS_GetAction(eCal;vAction)
  If(vAction=0) `did the user select a day or days?
    `get the events on this day or days
    CS_GetSellItems(eCal;1;aIndex)
    `build an array with the event text for events on the selected days
    FS_ArrayIntrsect(aIndex;aEventText;aSelDayEvts)
  End if `vAction=0)
End if `During
  
```

Util_SetDate

Util_SetDate(dateToSet; month; day; year)

Parameter	Type	Description
<i>DateToSet</i>	date	date to be set
<i>Month</i>	integer	month number of the date
<i>Day</i>	integer	day number of the date
<i>Year</i>	integer	year of the date

Util_SetDate allows you to set a 4D date variable in an international aware way. If you are trying to specify a date using variables for month, day and year, there is no way to know what order dates are specified. **Util_SetDate** lets you set a date consistently regardless of the system software version that is being used.

DateToSet — This parameter returns the date specified by *Month*, *Day*, and *Year*.

Month — Integer. Use this parameter to specify the month of the date you wish to set.

Day — Integer. Use this parameter to specify the day of the date you wish to set.

Year — Integer. Use this parameter to specify the year of the date you wish to set.

Example:

```

vDate := !12/1/93! `Could be Dec 1 or Jan 12 depending on country
Util_SetDate(vDate;12;1;1993) ` guaranteed to be Dec 1, 1993
  
```

Area_SetEnable

Area_SetEnable(AreaName; Enabled)

Parameter	Type	Description
<i>AreaName</i>	longint	name of plug-in area object on layout
<i>Enabled</i>	integer	enable or disable object

Area_SetEnable allows you to procedurally enable/disable any of the plug-in area objects in the CalendarSet package, including the CalendarSet object, and all the popup objects.

Enabled — Integer. Use this parameter to specify whether you wish to enable or disable *AreaName*.

0	disable
1	enable

A disabled plug-in object will be “grayed out” and will not accept clicks. After calling **Area_SetEnable** in the during phase of a layout, you must call **Area_Refresh** ([page 88](#)) to redraw the object.

Examples:

```
`disable the date popup eInvDate
Area_SetEnable (eInvDate;0)
`enable the color popup eColor
Area_SetEnable (eColor;1)
```

Area_Refresh

Area_Refresh(AreaName)

Parameter	Type	Description
<i>AreaName</i>	longint	name of plug-in area object on layout

Area_Refresh will redraw any of the plug-in area objects in the CalendarSet package, using all of the settings that were changed with any of the supplementary plug-ins. Use this command anytime that you want a calendar object, icon popup object, or color popup object redrawn. This is necessary when modifying an array displayed by a calendar or other object type, or modifying a configuration option.

You should only use this command in the **During** phase of the execution cycle.

Example:

CalendarSet Utility Commands

`after adding an item to the arrays being displayed in the calendar
`object eCalendar, update the object

AddCalendarItem(vDate;vText)

Area_Refresh(eCalendar)

CalendarSet in an plug-in Window

Using CalendarSet in an plug-in Window

You can display CalendarSet in an independent, resizable window. This type of window is created using the 4D **plug-in WINDOW** command. The syntax of the command is as follows:

plug-in WINDOW(*left;top;right;bottom;type;title;area*) → *ReferenceID*

Parameter	Type	Description
<i>left</i>	integer	pixel distance from left screen edge to left window edge
<i>top</i>	integer	pixel distance from top screen edge to top window edge
<i>right</i>	integer	pixel distance from left screen edge to right window edge
<i>bottom</i>	integer	pixel distance from top screen edge to bottom window edge
<i>type</i>	integer	window type
<i>title</i>	string	Window title, displayed in titlebar
<i>area</i>	longint	plug-in object name — use _CS_Area for CalendarSet
<i>ReferenceID</i>	longint	ID of plug-in area, used by CalendarSet commands

*For more information on plug-in windows, refer to the 4D reference manual discussion of the **plug-in WINDOW** command.*

After opening the plug-in window, you then issue CalendarSet commands to configure the format and events you wish to display.

You can detect user actions on the CalendarSet object, but since there is no script or layout procedure, a “callback” procedure is used. This is a procedure you specify with **CS_SetCallback** ([page 58](#)), which CalendarSet will execute when the user clicks or does a drag action on the object.

Example:

```

`display all items for this month from the appointments file
C_DATE(vDay1ThisMth;vDay1NextMth)
C_INTEGER($ThisMonth)
$ThisMonth:=Month of(Current date)
`See “Util_SetDate” on pag e87
Util_SetDate (vDay1ThisMth;$ThisMonth;1;Year of(Current date))
if ($ThisMonth<12)
  Util_SetDate (vDay1NextMth;$ThisMonth+1;1;Year of(Current date))
Else
  Util_SetDate (vDay1NextMth;1;1;Year of(Current date)+1)
End if
query([Appts];[Appts]Appt Date>=vDay1ThisMth;*) `find the appointments for
this month
query([Appts]; & ;[Appts]Appt Date<vDay1NextMth)
`now load the appointment data into arrays

```

CalendarSet in an plug-in Window

```
`Note that each appointment item has a field for the date, item text,
`font, size, style, and color. This makes loading the data and displaying it
`very easy to accomplish.
SELECTION TO ARRAY([Appts]Appt Date;aDates:[Appts]Item;alt-
ems:[Appts]Item Font;
aFonts:[Appts]Size;aSizes:[Appts]Styles;aStyles:[Appts]Color;
aColors)
`and display the data in the CalendarSet plug-in area named "eMonth"

`open a CalendarSet plug-in window
```

```
eMonth:=plug-in WINDOW(10;50;400;450;4;"plug-in Window Exam-
ple";"_CS_Area")
```

```
`setup the CalendarSet area
CS_SetArray (eMonth;"aDates";"altems";"aFonts";"aSizes";"aStyles";"aColors")
`highlight number only, multiple cells (contiguous), grayed cells with info,
`month prefix, sunday start, extend frame
CS_Options (eMonth;2;1;2;1;0;1)
`allow event selection, word-wrap event text, mark events with bullet
CS_SetEventOpts(eMonth;1;1;1)
`allow banner selection, allow banner resizing
CS_SetBanrOpts(eMonth;1;1)
`grid color blue, background color very light gray
CS_SetCalColor(eMonth;-5;241)
```

```
CS_SetCallback(eMonth;"HandleCalClick")
```

```


---


`GLOBAL PROCEDURE HandleCalClick
C_LONGINT($1;$CalArea) ` $1 = id of CalendarSet object
C_INTEGER($2;$Action) ` $2 = type of user action
$CalArea:=$1 `reassign the passed parameters for better readability
$Action:=$2
```

Case of

```
:( $Action=1) `user click on a day
    CS_GetSelect($CalArea;vDateSt;vDateEn;vContig;aDays)
    ModDayInfo ($CalArea;vDateSt) `add/modify the info for that day
:( $Action=2) `user click in plug-in window close box
    SaveCalendar `save any changes
:( $Action=3) `clicked on an event or banner

:( $Action=4) `dragged an event or banner

:( $Action=5) `resized a banner
```

End case

CalendarSet and Printing

You can easily print a layout with a CalendarSet object, using the commands discussed in [“Configuration Commands” on page 23](#).

The layout must be printed using **PRINT SELECTION** or **PRINT RECORD**. 4D does not support printing plug-in objects using the **PRINT LAYOUT** command.

The CalendarSet commands should be executed in the **Before** phase of the layout procedure or CalendarSet object script. If you use the CalendarSet object script, be sure to turn the object’s “Script Only if Modified” checkbox off, or the script will not run.

There are typically two ways the you may print a CalendarSet object:

- ◆ One CalendarSet object, containing one or more event records.
- ◆ One CalendarSet object for each record of a selection, with each CalendarSet object containing zero, one or more event records.

For both scenarios, you should remember that **PRINT SELECTION** will print the Detail area of the layout once for each record of the current selection in the file (or just once, if using **PRINT RECORD**). Therefore, if you only wish to print one CalendarSet object, with one or more event records, there must be only one record in the current selection for the file containing the layout you are printing.

CalendarSet Examples

The examples in this section are designed to provide an overview of the use of CalendarSet and the basic commands. The examples are included on your CalendarSet disk, to allow experimentation with the commands and parameters.

These examples use CalendarSet commands to configure the behavior of an object, rather than the Advanced Properties Dialog. You can reduce the programming effort by using the Advanced Properties Dialog. *Please read the section [“Configuring CalendarSet Using the Advanced Properties Dialog” on page23 for more information.](#)*

You may also wish to examine the non-compiled version of the CalendarSet demo, for more examples on the various CalendarSet capabilities.

Example 1 — Create a Simple Calendar

For this example, we will create a simple layout containing a single CalendarSet object. When the user clicks on a day, we will put the selected date into a variable called *vSelDate*.

First we need to create the layout and draw the CalendarSet plug-in object. We'll name the object *eCalendar*.

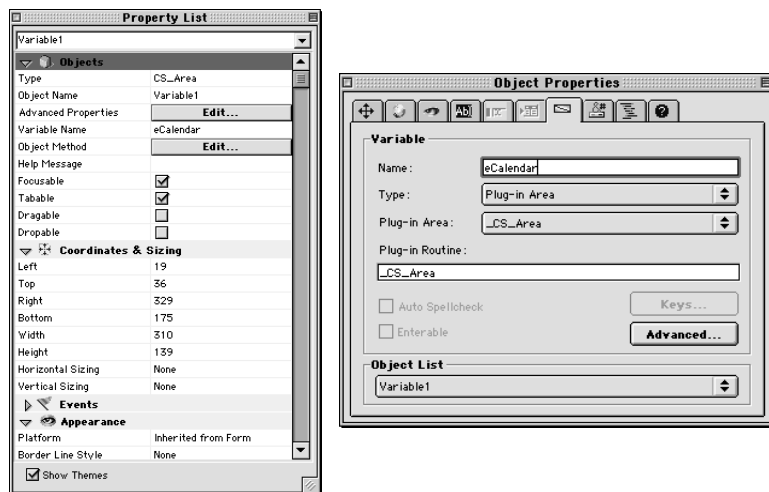


Figure 17a & b

Our layout now looks something like this:

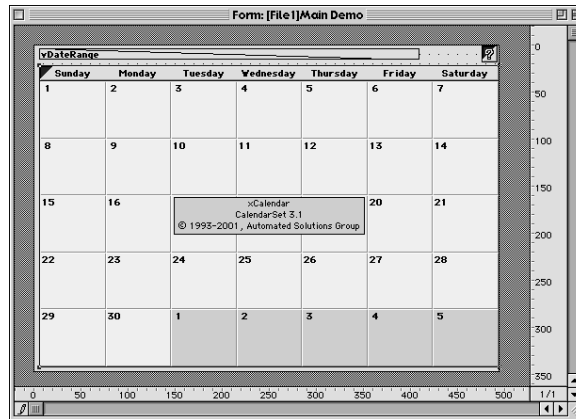


Figure 18

The script for the CalendarSet object is:

```

If(During)
  CS_GetSelect(eCalendar;vSelDate;vSelDateE;isContig)
End if

```

The layout will appear like this in the User or Runtime environment:



Figure 19

Notice that we've used the default configuration options, and the only programming required is the command to determine what day the user has selected.

Example 2 — Display Database Data in a Calendar

Modify the previous example to display information from the [Cal

Items] file on the calendar. The file structure for this example is:

Cal Example	
StartDate	D
EndDate	D
ItemType	I
Description	A80
Font	A40
Size	I
FaceOrResID	I
ForeColor	I
BackColor	I
SeqNo	L

Figure 20

The modified script for the CalendarSet object is:

Case of

: (Before)

vSelDate:=!00/00/00!

CS_SetRange(eCalendar;!02/01/95!;!02/28/95!)

query([Cal Items];[Cal Items]Item Type=0)pp Items of type 0 are text items for this demo

SELECTION TO ARRAY([Cal Items]Start Date;vTextDate;[Cal Items]Description;vTextDesc;[Cal Items]Font;vTextFont;[Cal Items]Size;vTextSize;[Cal Items]Style;vTextStyle;[Cal Items]ForeColor;vTextColor)

CS_SetArray(eCalendar;"vTextDate";"vTextDesc";"vTextFont";"vTextSize";"vTextStyle";"vTextColor")

: (During)

CS_GetSelect(eCalendar;vSelDate;vSelDateE;isContig)

End case

The CalendarSet object now appears in the User or Runtime environment like this:

Example 2						
Sunday	Monday	Tuesday	Wednesday	Thursday	Friday	Saturday
1	2	3	4	5	6	
7 Mow the Lawn At Star Game	8	9	10	11	12	13
14	15	16	17	18	19	20
21 Buck & Betty at	22	23	24	25	26	27
28 Author's Bir	1	2	3	4	5	6

Selected Date: 02/17/93 Done

Figure 21

Example 3 — Displaying Banners on a Calendar

We'll change the previous example to also display banner items. We can use the same CalendarSet object we created in the previous examples, and just modify the script.

Our new script is:

Case of

: (Before)

```
vSelDate:=!00/00/00!
```

```
`set the range of the calendar to be Feb 1993
```

```
CS_SetRange(eCalendar;!02/01/93;!02/28/93!)
```

```
`Items of type 0 are text items
```

```
þ`Find all of the text items
```

```
query([Cal Items];[Cal Items]Item Type=0)
```

```
`load the arrays with the data
```

```
SELECTION TO ARRAY([Cal Items]Start Date;vTextDate;[Cal Items]Description;vTextDesc;[Cal Items]Font;vTextFont;[Cal Items]Size;vTextSize;[Cal Items]Style;vTextStyle;[Cal Items]ForeColor;vTextColor)
```

```
`pass the arrays to CalendarSet
```

```
CS_SetArray(eCalendar;"vTextDate";"vTextDesc";"vTextFont";"vTextSize";"vTextStyle";"vTextColor")
```

```
`Items of type 0 are Banner items
```

```
`Find all of the Banner items
```

```
query([Cal Items];[Cal Items]Item Type=1)
```

```
`pass the arrays to CalendarSet
```

```
SELECTION TO ARRAY([Cal Items]Start Date;vBanrDateS;[Cal Items]EndDate;vBanrDateE;[Cal Items]Description;vBanrDesc;[Cal Items]Font;vBanrFont;[Cal Items]Size;vBanrSize;[Cal Items]Style;vBanrStyle;[Cal Items]ForeColor;vBanrColorF;[Cal Items]BackColor;vBanrColorB)
```

```
`pass the arrays to CalendarSet
```

```
CS_SetBanrArray(eCalendar;"vBanrDateS";"vBanrDateE"; "vBanrDesc";"vBanrFont";"vBanrSize";"vBanrStyle";"vBanrColorF";"vBanrColorB")
```

: (During)

```
CS_GetSelect(eCalendar;vSelDate;vSelDateE;isContig;selection)
```

End case

Our layout now looks like this to the user:

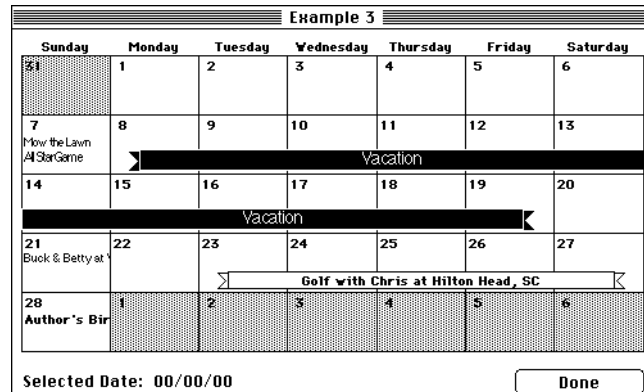


Figure 22

Example 4 — Responding to User Actions

In the previous examples, we've gradually added capabilities to the calendar object, so that we can display both text and banner items, and identify the day clicked by a user. For this example, we'll implement two scrolling areas which show information about the day or days selected. The first scrolling area will show the days selected. The second scrolling area will show any items displayed on the selected day or days.

Here's the modified script:

Case of

: (Before)

vSelDate:=!00/00/00!

vSelDateE:=!00/00/00!

`set the range of the calendar to be Feb 1993

CS_SetRange(eCalendar;!02/01/93;!02/28/93!)

`Items of type 0 are text items

`Find all of the text items

query([Cal Items];[Cal Items]Item Type=0)

`load the arrays with the data

SELECTION TO ARRAY([Cal Items]Start Date;vTextDate;[Cal Items]Description;vTextDesc;[Cal Items]Font;vTextFont;[Cal Items]Size;vTextSize;[Cal Items]Style;vTextStyle;[Cal Items]ForeColor;vTextColor)

`pass the arrays to CalendarSet

CS_SetArray(eCalendar;"vTextDate";"vTextDesc";"vTextFont";"vTextSize";"vTextStyle";"vTextColor")

`Items of type 0 are Banner items

`Find all of the Banner items

query([Cal Items];[Cal Items]Item Type=1)

`load the arrays with the data

SELECTION TO ARRAY([Cal Items]Start Date;vBanrDateS;[Cal Items]EndDate;vBanrDateE;[Cal Items]Description;vBanrDesc;[Cal Items]Font;vBanrFont;[Cal Items]Size;vBanrSize;[Cal

```

Items]Style;vBanrStyle;[Cal Items]ForeColor;vBanrColorF;[Cal
Items]BackColor;vBanrColorB)
`pass the arrays to CalendarSet
CS_SetBanrArray(eCalendar;"vBanrDateS";"vBanrDateE"; "vBanr-
Desc";"vBanrFont";"vBanrSize";"vBanrStyle";"vBanrCol-
orF";"vBanrColorB")
p`Set the options to display entire highlight, allow discontiguous selection, do
`not show day number for unused days, and display month names
CS_Options(eCalendar;1;2;1;1)
: (During)
CS_GetSelect(eCalendar;vSelDate;vSelDateE;isContig;selection)
`get an index of the selected items
CS_GetSelltems(eCalendar;1;vSelltems)
p`create an array of the selected items only, using the index
FS_ArrayIntrsect(vSelltems;vTextDesc;vSelText)
End case

```

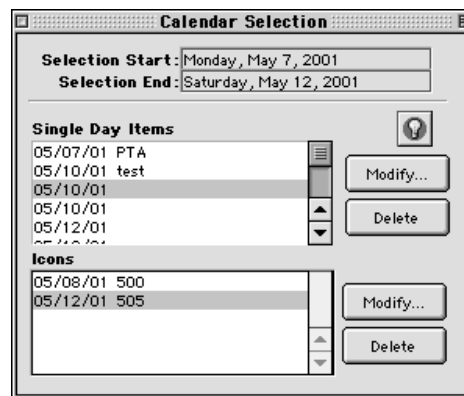


Figure 23

Example 5 — Dragging Events within a CalendarSet Object

Once again we'll modify the previous example, to allow events to be dragged within the CalendarSet area. Since CalendarSet will automatically update the date arrays for events when they are dragged within an object, we don't have to handle the user actions (although we could do this if needed). We use **CS_SetDrgSrc** ([page 64](#)) and **CS_SetDrgDst** ([page 65](#)) to restrict event dragging to be within the object.

Here's the modified script:

```

Case of
: (Before)
vSelDate:=!00/00/00!
vSelDateE:=!00/00/00!

```

```

    `set the range of the calendar to be Feb 1993
CS_SetRange(eCalendar;!02/01/93;!02/28/93!)
    `Items of type 0 are text items
    `Find all of the text items
query([Cal Items];[Cal Items]Item Type=0)
    `load the arrays with the data
SELECTION TO ARRAY([Cal Items]Start Date;vTextDate;[Cal Items]Description;vTextDesc;[Cal Items]Font;vTextFont;[Cal Items]Size;vTextSize;[Cal Items]Style;vTextStyle;[Cal Items]ForeColor;vTextColor)
    `pass the arrays to CalendarSet
CS_SetArray(eCalendar;"vTextDate";"vTextDesc";"vTextFont";"vTextSize";"vTextStyle";"vTextColor")
    `Items of type 0 are Banner items
    `Find all of the Banner items
query([Cal Items];[Cal Items]Item Type=1)
    `load the arrays with the data
SELECTION TO ARRAY([Cal Items]Start Date;vBanrDateS;[Cal Items]EndDate;vBanrDateE;[Cal Items]Description;vBanrDesc;[Cal Items]Font;vBanrFont;[Cal Items]Size;vBanrSize;[Cal Items]Style;vBanrStyle;[Cal Items]ForeColor;vBanrColorF;[Cal Items]BackColor;vBanrColorB)
    `pass the arrays to CalendarSet
CS_SetBanrArray(eCalendar;"vBanrDateS";"vBanrDateE"; "vBanrDesc";"vBanrFont";"vBanrSize";"vBanrStyle";"vBanrColorF";"vBanrColorB")
    p`Set the options to display entire highlight, allow discontinuous selection, do
    `not show day number for unused days, and display month names
CS_Options(eCalendar;1;2;1;1)

    ` enable dragging events to days within this area
    vSelfStr:=String(eCalendar) ` creates a unique code that only allows dragging
    within this area
CS_SetDrgSrc(eCalendar;2;vSelfStr) ` event data type for source
CS_SetDrgDst(eCalendar;1;vSelfStr) ` day data type for destination

: (During)
CS_GetSelect(eCalendar;vSelDate;vSelDateE;isContig;selection)
    `get an index of the selected items
CS_GetSelltems(eCalendar;1;vSelltems)
    p`create an array of the selected items only, using the index
FS_ArrayIntrsect(vSelltems;vTextDesc;vSelText)
End case

```

Example 6 — Dragging Events to another CalendarSet Object on the Same Layout

Our previous example only allowed events to be dragged within the CalendarSet area. Since CalendarSet automatically updates the date arrays for events when they are dragged within an object, we didn't have to handle the user actions. We'll extend our example to show how to handle drags between CalendarSet objects on the same layout. The two objects are named eCalendar and eOther.

When the user drags an event from eCalendar to eOther, we'll "copy" that event into the arrays displayed in eOther.

We'll just show the script for eCalendar.

Case of

: (Before)

vSelDate:=!00/00/00!

vSelDateE:=!00/00/00!

`set the range of the calendar to be Feb 1993

CS_SetRange(eCalendar;!02/01/93;!02/28/93!)

`Items of type 0 are text items

`Find all of the text items

query([Cal Items];[Cal Items]Item Type=0)

`load the arrays with the data

SELECTION TO ARRAY([Cal Items]Start Date;vTextDate;[Cal Items]Description;vTextDesc;[Cal Items]Font;vTextFont;[Cal Items]Size;vTextSize;[Cal Items]Style;vTextStyle;[Cal Items]ForeColor;vTextColor)

`pass the arrays to CalendarSet

CS_SetArray(eCalendar;"vTextDate";"vTextDesc";"vTextFont";"vTextSize";"vTextStyle";"vTextColor")

`Items of type 0 are Banner items

`Find all of the Banner items

query([Cal Items];[Cal Items]Item Type=1)

`load the arrays with the data

SELECTION TO ARRAY([Cal Items]Start Date;vBanrDateS;[Cal Items]EndDate;vBanrDateE;[Cal Items]Description;vBanrDesc;[Cal Items]Font;vBanrFont;[Cal Items]Size;vBanrSize;[Cal Items]Style;vBanrStyle;[Cal Items]ForeColor;vBanrColorF;[Cal Items]BackColor;vBanrColorB)

`pass the arrays to CalendarSet

CS_SetBanrArray(eCalendar;"vBanrDateS";"vBanrDateE"; "vBanrDesc";"vBanrFont";"vBanrSize";"vBanrStyle";"vBanrColorF";"vBanrColorB")

þ`Set the options to display entire highlight, allow discontinuous selection, do not show day number for unused days, and display month names

CS_Options(eCalendar;1;2;1;1)

`enable dragging events to days within this area

vSelfStr:=**String**(eCalendar) `creates a unique code that allows dragging within this area

vOtherStr:=**String**(eOther) `creates a unique code that allows dragging to the other area

CS_SetDrgSrc(eCalendar;2;vSelfStr;vOtherStr) `event data type for source

CS_SetDrgDst(eCalendar;1;vSelfStr;vOtherStr) `day data type for destination

: (During)

CS_GetAction(eCalendar;vActionCode)

Case of

:(vActionCode=4) `user dragged an event

CS_GetDrgSrcEvt(eCalendar;vEvtType;vEvtIndex;vDate)

CS_GetDrgArea(eCalendar;vDestArea;vDestProclD)

If(vDestArea=eOther)

CS_GetDrgDstTyp(eOther;vDestType)

If(vDestType=1) `dragged onto a day?

```

        CS_GetDrgDstDay(eOther;vDestDate)
        ADD EVENT(vEvtIndex;vDestDate)
    End if
End if
End case
End case

```

Example 8 — Dragging Events to a CalendarSet Object on a Different Layout

We'll modify the previous example to show how to handle dragging events to a CalendarSet object on a different layout. Instead of the eOther object being on the same layout, we'll put it on a different layout.

We have to use the 4D **CALL PROCESS** and **Outside call** commands to assist with the implementation. When we detect a drag from eCalendar to eOther, our eCalendar script will send a message to eOther's layout procedure, which will then update the arrays.

```

Case of
: (Before)
    vSelDate:=!00/00/00!
    vSelDateE:=!00/00/00!
    `set the range of the calendar to be Feb 1993
    CS_SetRange(eCalendar;!02/01/93!;!02/28/93!)
    `Items of type 0 are text items
    `Find all of the text items
    query([Cal Items];[Cal Items]Item Type=0)
    `load the arrays with the data
    SELECTION TO ARRAY([Cal Items]Start Date;vTextDate;[Cal Items]Description;vTextDesc;[Cal Items]Font;vTextFont;[Cal Items]Size;vTextSize;[Cal Items]Style;vTextStyle;[Cal Items]ForeColor;vTextColor)
    `pass the arrays to CalendarSet
    CS_SetArray(eCalendar;"vTextDate";"vTextDesc";"vTextFont";"vTextSize";"vTextStyle";"vTextColor")
    `Items of type 0 are Banner items
    `Find all of the Banner items
    query([Cal Items];[Cal Items]Item Type=1)
    `load the arrays with the data
    SELECTION TO ARRAY([Cal Items]Start Date;vBanrDateS;[Cal Items]EndDate;vBanrDateE;[Cal Items]Description;vBanrDesc;[Cal Items]Font;vBanrFont;[Cal Items]Size;vBanrSize;[Cal Items]Style;vBanrStyle;[Cal Items]ForeColor;vBanrColorF;[Cal Items]BackColor;vBanrColorB)
    `pass the arrays to CalendarSet
    CS_SetBanrArray(eCalendar;"vBanrDateS";"vBanrDateE"; "vBanrDesc";"vBanrFont";"vBanrSize";"vBanrStyle";"vBanrColorF";"vBanrColorB")
    p`Set the options to display entire highlight, allow discontinuous selection, do
    `not show day number for unused days, and display month names
    CS_Options(eCalendar;1;2;1;1)

```

```

    ` enable dragging events to days within this area
CS_SetDrgSrc(eCalendar;2;"DragExample") ` event data type for source
CS_SetDrgDst(eCalendar;1;"DragExample") ` day data type for destination

: (During)
CS_GetAction(eCalendar;vActionCode)
Case of
:(vActionCode=4) `user dragged an event
    CS_GetDrgSrcEvt(eCalendar;vEvtType;PvEvtIndex;vDate)
    CS_GetDrgArea(eCalendar;vDestArea;vDestProclD)
    If(vDestProclD#Current process) `was the drag to a different window?
        CALL PROCESS(vDestProclD)
    End if
End case
End case

`
`-----
`Other script
Case of
:(Before)
    `we'll not show the standard setup stuff, and just show the dragging-specific
        code
    vSelfStr:="OurFirstArea"
    vOtherStr:="OurOtherArea"
    CS_SetDrgSrc(eOther;2;"DragExample") ` event data type for source
    CS_SetDrgDst(eOther;1;"DragExample") ` day data type for destination
:(During)
    `the same stuff here as the eCalendar script
:(Outside call) `this is how we get the message
    CS_GetDrgDstTyp(eOther;vDestType)
    If(vDestType=1) `dragged onto a day?
        CS_GetDrgDstDay(eOther;vDestDate)
        ADD EVENT(PvEvtIndex;vDestDate)
    End if
End case

```

Example 9 — Dragging from AreaList Pro to CalendarSet

This is an example of dragging a row from an AreaList Pro object in one process to a CalendarSet object in another process.

This example assumes that the AreaList Pro object has already been set up for dragging a row using **AL_SetDrgSrc** ([page 64](#)) and that the CalendarSet object has already been set up for receiving a drag using **CS_SetDrgDst** ([page 65](#)).

First, the script of the AreaList Pro object:

```

Case of
:(During)
Case of
:(ALProEvt = -5) ` a row was dragged

```

AL_GetDrgSrcRow(Self»;vSrcRow) ` get the row that was dragged

`get the area and process that was dragged to

AL_GetDrgArea(Self»;vDestArea;vDestProclD)

If(vDestProclD # **Current process**)

`store the destination area and text in interprocess variables

pdestArea := vDestArea

pdestText := array1{vSrcRow}+" "+ array2{vSrcRow}

`call the other process to update the destination area

CALL PROCESS(vDestProclD)

End if

End Case

End Case

Next, the layout procedure of the layout containing the CalendarSet object:

Case of

:(**Outside call**)

Case of

:(pdestArea = eCalArea)

`add an element to both the date and text arrays

INSERT ELEMENT(aCSDate;1000)

INSERT ELEMENT(aCSText;1000)

`get the date of the day that was dragged to

CS_GetDrgDstDay(eCalArea;vTempDate)

`update the arrays with the date and the text that was set in the other
`process

aCSDate{Size of array(aCSDate)}:= vTempDate

aCSText{Size of array(aCSText)}:= pdestText

Area_Refresh(eCalArea) ` update the area

End case

End case

Troubleshooting

CalendarSet is not being updated properly

- ◆ Make sure that you are calling **Area_Refresh** ([page 88](#)) whenever you modify any of the displayed arrays.
- ◆ If you modify any array which was passed to CalendarSet using **CS_SetArray** ([page 35](#)), **CS_SetBannerArray** ([page 38](#)), or **CS_SetIconArray** ([page 40](#)), you must *not* reissue any of those commands — simply use **Area_Refresh** ([page 88](#)) to tell CalendarSet to use the new data in the arrays.
- ◆ Make sure that you do not have two active CalendarSet objects with the same name.

CalendarSet does not respond to single or double-clicks

- ◆ [See “Responding to User Actions on a CalendarSet Object” on page 52.](#)
- ◆ A CalendarSet object can not be displayed on an input layout entered via the **DISPLAY SELECTION** command.

CalendarSet™ Demo

The CalendarSet Demo database allows the various features of CalendarSet to be experimented with. The structure is provided in both compiled and non-compiled form. Feel free to use any of the procedures in your own code; there are several useful routines included.

Command Reference — by Chapter

Configuration Commands 23

<i>CS_Register</i> (SerialNumOne; SerialNumTwo) → ResultCode.....	32
<i>CS_SetRange</i> (area name;StartDate;EndDate).....	34
<i>CS_SetSelect</i> (area name;StartDate;EndDate;Contiguous;DaysArray).....	34
<i>CS_SetArray</i> (area name;DateArray;TextArray; FontArray; SizeArray; StyleArray; ColorArray).....	35
<i>CS_SetBannerArray</i> (area name; StartDateArray; EndDateArray; TextArray; FontArray; SizeArray; StyleArray; ForeColorArray; BackColorArray).....	38
<i>CS_SetIconArray</i> (area name; DateArray; IconArray; SizeArray).....	40
<i>CS_Options</i> (AreaName;DayHighlightMode;DaySelectionType;UnusedInfo;MonthPrefix; FirstDayOfWeek;ExtendFrame).....	41
<i>CS_SetEventOpts</i> (AreaName; AllowEventSelect; ShowWordWrap; EventMarker).....	42
<i>CS_SetBannerOpts</i> (AreaName; AllowBannerSelect; AllowBannerResize).....	43
<i>CS_SetCalColor</i> (AreaName; CalForeColor; CalBackColor).....	44
<i>CS_SetEvtSelect</i> (AreaName; EventType; EventIndex).....	45
<i>CS_HideDays</i> (area name; HideSunday; HideMonday; HideTuesday; HideWednesday; HideThursday; HideFriday; HideSaturday).....	46
<i>CS_FontDefaults</i> (area name; Selector; Font; Size; Style; ForeColor; BackColor).....	47
<i>CS_SetDayStyle</i> (area name; TargetDate; DayFont; DaySize; DayStyle; DayColor; ClearOthers).....	49
<i>CS_SetDayStyleB</i> (area name; DateArray; DayFont; DaySize; DayFace; DayColor; ClearOthers).....	50
<i>CS_SetSizeOpts</i> (AreaName; AllowResize; MinWidth; MinHeight; MaxWidth; MaxHeight).....	51

User Action Commands 52

<i>CS_GetSelect</i> (area name;StartDate;EndDate;Contiguous;DaysArray).....	52
<i>CS_GetAction</i> (AreaName;ActionCode).....	53
<i>CS_GetEvtSelect</i> (AreaName; EventType; EventIndex).....	54
<i>CS_GetSelItems</i> (area name; ItemType; SelectedItems).....	55
<i>CS_LastClick</i> (area name; DateClicked; WasDouble; Modifiers).....	56
<i>CS_GetResizBanner</i> (AreaName; BannerIndex).....	57
<i>CS_SetCallback</i> (area name; ProcToExecute).....	58
<i>CS_DoResize</i> (AreaName).....	59

CalendarSet Dragging Commands..... 60

<i>CS_DragMgrAvail</i> (IsDragMgrPresent).....	63
<i>CS_SetDrgSrc</i> (AreaName; SourceDataType; SrcCode1; SrcCode2; ... ; SrcCode10).....	64
<i>CS_SetDrgDst</i> (AreaName; DestDataType; DstCode1; DstCode2; ... ; DstCode10).....	65
<i>CS_GetDrgSrcEvt</i> (AreaName; EventType; EventIndex; StartDate).....	67
<i>CS_GetDrgArea</i> (AreaName; DestArea; DestProcessID).....	68
<i>CS_GetDrgDstTyp</i> (AreaName;DestDataType).....	69
<i>CS_GetDrgDstDay</i> (AreaName; DestDate).....	70
<i>CS_GetDrgDstEvt</i> (AreaName; EventType; EventIndex).....	71

Popup Commands 74

<i>MM_Options</i> (AreaName; HidePopArrow).....	76
<i>MM_SetDate</i> (area name; DateToSet).....	76
<i>MM_GetDate</i> (area name; DateToGet).....	77
<i>TM_Options</i> (AreaName; HidePopArrow).....	77
<i>TM_SetTime</i> (area name;TimeToSet).....	78
<i>TM_GetTime</i> (area name;TimeToGet).....	79
<i>CM_SetColor</i> (area name; ColorToSet).....	80
<i>CM_GetColor</i> (area name; ColorToGet).....	81
<i>IM_SetArray</i> (area name; ArrayName; IconSize; NumColumns).....	82
<i>IM_GetSelect</i> (area name; CurrentSelectedItem).....	82
<i>IM_SetSelect</i> (area name; CurrentSelectedItem).....	83

CalendarSet Utility Commands 84

<i>CS_GetDayName</i> (DayNumber; Abbreviated; DayName).....	85
<i>CS_GetMonthName</i> (MonthNumber; Abbreviated; MonthName).....	85
<i>FS_ArrayIntrscct</i> (CompareArray; SourceArray; DestArray).....	86
<i>Util_SetDate</i> (dateToSet; month; day; year).....	87
<i>Area_SetEnable</i> (AreaName; Enabled).....	88
<i>Area_Refresh</i> (AreaName).....	88

CalendarSet in an plug-in Window..... 90

plug-in WINDOW(left;top;right;bottom;type;title;area) → ReferenceID.....	90
--	----

Command Reference — Alphabetical

Area_Refresh (AreaName)	88
Area_SetEnable (AreaName; Enabled)	88
CM_GetColor (area name; ColorToGet)	81
CM_SetColor (area name; ColorToSet)	80
CS_DoResize (AreaName)	59
CS_DragMgrAvail (IsDragMgrPresent)	63
CS_FontDefaults (area name; Selector; Font; Size; Style; ForeColor; BackColor)	47
CS_GetAction (AreaName;ActionCode)	53
CS_GetDayName (DayNumber; Abbreviated; DayName)	85
CS_GetDrgArea (AreaName; DestArea; DestProcessID)	68
CS_GetDrgDstDay (AreaName; DestDate)	70
CS_GetDrgDstEvt (AreaName; EventType; EventIndex)	71
CS_GetDrgDstTyp (AreaName; DestDataType)	69
CS_GetDrgSrcEvt (AreaName; EventType; EventIndex; StartDate)	67
CS_GetEvtSelect (AreaName; EventType; EventIndex)	54
CS_GetMonthName (MonthNumber; Abbreviated; MonthName)	85
CS_GetResizBanrr (AreaName; BannerIndex)	57
CS_GetSelect (area name;StartDate;EndDate;Contiguous;DaysArray)	52
CS_GetSellItems (area name; ItemType; SelectedItems)	55
CS_HideDays (area name; HideSunday; HideMonday; HideTuesday; HideWednesday; HideThursday; HideFriday; HideSaturday)	46
CS_LastClick (area name; DateClicked; WasDouble; Modifiers)	56
CS_Options (AreaName;DayHighlightMode;DaySelectionType;UnusedInfo;MonthPrefix; FirstDayOfWeek;ExtendFrame)	41
CS_Register (SerialNumOne; SerialNumTwo) → ResultCode	32
CS_SetArray (area name;DateArray;TextArray; FontArray; SizeArray; StyleArray; ColorArray)	35
CS_SetBanrrArray (area name; StartDateArray; EndDateArray; TextArray; FontArray; SizeArray; StyleArray; ForeColorArray; BackColorArray)	38
CS_SetBanrrOpts (AreaName; AllowBannerSelect; AllowBannerResize)	43
CS_SetCalColor (AreaName; CalForeColor; CalBackColor)	44
CS_SetCallback (area name; ProcToExecute)	58
CS_SetDayStyle (area name; TargetDate; DayFont; DaySize; DayStyle; DayColor; ClearOthers)	49
CS_SetDayStyleB (area name; DateArray; DayFont; DaySize; DayFace; DayColor; ClearOthers)	50
CS_SetDrgDst (AreaName; DestDataType; DstCode1; DstCode2; ... ; DstCode10)	65
CS_SetDrgSrc (AreaName; SourceDataType; SrcCode1; SrcCode2; ... ; SrcCode10)	64
CS_SetEventOpts (AreaName; AllowEventSelect; ShowWordWrap; EventMarker)	42
CS_SetEvtSelect (AreaName; EventType; EventIndex)	45
CS_SetIconArray (area name; DateArray; IconArray; SizeArray)	40
CS_SetRange (area name;StartDate;EndDate)	34
CS_SetSelect (area name;StartDate;EndDate;Contiguous;DaysArray)	34
CS_SetSizeOpts (AreaName; AllowResize; MinWidth; MinHeight; MaxWidth; MaxHeight)	51
FS_ArrayIntrct (CompareArray; SourceArray; DestArray)	86
IM_GetSelect (area name; CurrentSelectedItem)	82
IM_SetArray (area name; ArrayName; IconSize; NumColumns)	82
IM_SetSelect (area name; CurrentSelectedItem)	83
MM_GetDate (area name; DateToGet)	77
MM_Options (AreaName; HidePopArrow)	76
MM_SetDate (area name; DateToSet)	76
plug-in WINDOW (left;top;right;bottom;type;title;area) → ReferenceID	90
TM_GetTime (area name;TimeToGet)	79
TM_Options (AreaName; HidePopArrow)	77
TM_SetTime (area name;TimeToSet)	78
Util_SetDate (dateToSet; month; day; year)	87