CalendarSet User Manual

Version 4.0

e-Node

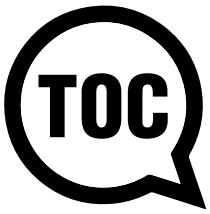
30 rue de la République

33150 Cenon

France

www.e-node.net

Manual revision: 5 Sept 2016



Contents

About CalendarSet

What is CalendarSet, and what can I do with it?

CalendarSet is an easy-to-use tool for implementing calendars on 4D 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.

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.

Technical Details

Compatibility Information

CalendarSet version 4 is compatible with 4D v13, v14, v15 and above, for both MacOS and Windows (including 32 bit and 64 bit servers). It requires MacOS 10.6.8 or higher and Windows XP SP2 or better.

You do not have to update your CalendarSet areas and code. The commands are still here and will work with CalendarSet version 4 with little or no change in your code. See the Upgrading from Previous versions section.

Technical Support

Technical support for CalendarSet is provided electronically via e-mail or our online support reporting system.

You are encouraged to use the online web forums.

Items that are new or modified in CalendarSet version 4 are displayed in pink (magenta) characters.

Note: many examples with their source code are available in the CalendarSet Demonstration database.

Installation

Installing the plugin

CalendarSet is provided as a bundle for both Windows and MacOS: there is just one version for both platforms. To install it, simply copy the file **CS.bundle** into your Plugins folder.

Plugins folders can be located in one of two locations:

- In the 4D application folder (4D or 4D Server). When plugins are installed in this location, they will be available to every database that is opened with that application.
- Next to the database structure file for your project: in this case, the plugin will only be available to that database. On MacOS, this means that the Plugins folder must be placed within the database package or folder. To open a package, ctrl-click on the package and choose **Show Package Contents** from the contextual menu.

Using CalendarSet in Demo mode

You can use CalendarSet in Demo mode for 20 minutes, after which time it will cease to work. When this becomes annoying, it's time to buy a license, which you can do on our website.

Licenses are either linked to the 4D product number, the workstation or the company name as described below.

Licensing

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

For current pricing, please see the ordering page on our website.

Definitions

- **Regular licenses** are used for applications that are opened with 4D Standalone or 4D SQL Desktop, or with 4D Server, either in interpreted or compiled mode (doesn't make a difference regarding plugin licensing).

  These can be either single user or server databases and they are linked to the 4D or 4D Server license: you need to provide the number returned by the "Copy" or "eMail" buttons from the plugin demonstration mode alert (this number is actually the 4D command **GET SERIAL INFORMATION** first parameter). This number is a negative long integer such as -1234567.

- **Merged licenses** are used for double-clickable applications built with 4D Volume Desktop (single user) or with 4D Server by means of the 4D Compiler module.

  These licenses are linked to the machine ID (single user workstation or server): you need to provide the number returned by the "Copy" or "eMail" buttons from the plugin demonstration mode alert (this number is calculated from the single user or server machine UUID). On 4D Server any remote client will return the server number. This number is a positive long integer such as 1234567.

In both cases the demonstration mode dialog will display the proper number according to the current setup (regular or merged) and the "Copy" and "eMail" buttons will use it as well.

Free updates

- **Regular licenses**. A new license will be supplied for free at any time (maximum once a year) if you change your 4D version or get a new 4D registration key for the same version, provided that your previous license match the current public version at exchange time. This rule applies whether you are already using the new version or not: just specify that you also want a key for the older version as well as the current one when you order an upgrade.
- **Merged licenses**. These licenses are independent from the 4D versions and product numbers. They will remain functional if you upgrade e.g. from 4D v14 to 4D v15 on the same machine (single user workstation or server).

  You'll only need to update a merged license if your machine or motherboard is replaced (a new license will be supplied for free in this case, provided that your previous license match the current public version at the exchange time), or if you install a paid upgrade of the plugin.

Note: if you are using several concurrent versions of 4D you will need one plugin license for each version.

License types

- **Single-user.** This license allows development (interpreted mode) or deployment (interpreted or compiled mode, including merged) of applications that are opened with 4D Standalone or 4D SQL Desktop or built with 4D Volume Desktop.
- **Server.** These licenses allow development (interpreted mode) or deployment (interpreted or compiled mode, including merged servers / remotes) on 4D Server with up to 10 users (“small server”), 11 to 20 users (“medium server”) or more (“large server”).
- **Unlimited Single User.** This license allows development (interpreted mode) or deployment (interpreted or compiled mode, including merged) on any number of 4D Standalone (or single user merged applications built with 4D Volume Desktop) that run your 4D application(s).

  It is a yearly license, which expires after the date when it is to be renewed. Expiration only affects interpreted mode. **Compiled applications using an obsolete license will never expire**.

  A single license key will unlock all setups on all compatible 4D versions and all versions of the plugin. The license key is linked to the developer/company name.

  This license allows deployment (selling new application licenses, updates or subscriptions) while the license is valid. **No new deployment may occur after expiry without a specific license** (merged or regular). End-users running deployments sold during the license validity period remain authorized without time limit, provided that they are no longer charged for the application using the plug-in (including maintenance or upgrades).
- **OEM.** This license allows development (interpreted mode) or deployment (interpreted or compiled mode, including merged) on any number of 4D Servers (any number of users), 4D Standalone or single user/remote merged instances that run your 4D application(s).

  It is a yearly license, under the exact same terms as the Unlimited Single User license described above, except that it also covers server deployments.
- **Unlimited OEM.** This license is a global OEM license, covering any combination of the plug-ins published by e-Node, including AreaList Pro, SuperReport Pro, PrintList Pro, CalendarSet and Internet ToolKit in all configurations.
- **Partner license.** This license matches 4D’s annual Partner subscription and covers all the plug-ins published by e-Node, including AreaList Pro, SuperReport Pro, PrintList Pro, CalendarSet and Internet ToolKit.

  For each product, a single registration key allows development (interpreted mode) or deployment (interpreted or compiled mode, except merged) on all 4D Standalones and 4D Servers (2 users) regardless of 4D product numbers, OS and versions. No merged applications.

  This is a yearly license, expiring on February 1st (same date as 4D Partner licenses). Expiration only affects interpreted mode. **Compiled applications using an obsolete license will never expire**.

Note: you don’t have to be a 4D Partner subscriber to subscribe to the e-Node Partner license.

Registering your CalendarSet License

Once you have purchased your license, you will receive a registration key. This code must be registered each time the database is started. There are four ways to register your license:

- using the Demo mode dialog “Register” button,
- though a text file,
- in your 4D code with a command,
- through the online automated registration system.

Yearly licenses such as Unlimited single user, OEM and Partner do not require any serial information or online registration. The only way to register these licenses is through the CS_Register command.

Quick and easy way

1. Put the following lines of code into your **On Startup** database method, with the license number that you received and your email address:

```
C_LONGINT ($result)
$result:=CS_Register ("yourLicenseKey";0;"youremail@something.xxx")  // 0 if successful
```

2. Make sure that the machine where the plugin will be used is connected to the Internet (single user workstation or in server mode the first remote client that will connect to the server).

3. Install your application.

4. The plugin will silently (no dialog) register itself.

5. You will receive an email with the final key issued and the IP address of the user site.

If the site has no Internet connection or if you want to use the plugin license system to help protect your own software copy, you can manage the final key registration yourself using one of the following methods.

The Demo mode dialog

Single user and server licenses require that you first send us the relevant information (serial or machine ID, see Definitions), unless you are using the Online registration system.

This is performed from the demonstration mode dialog.

The Demo mode dialog is displayed upon the first call to ITK (through a command).

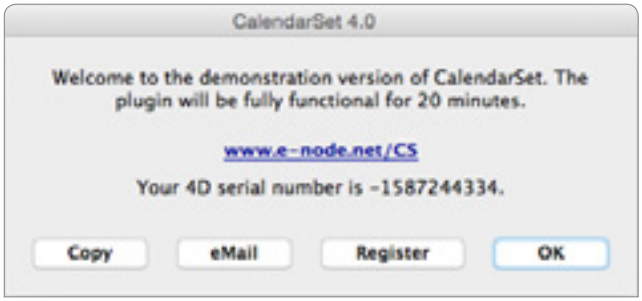
To trigger this display and enable your users to register without actually calling a command or setting up an area, pass an **empty** string to ***CS_Register*** and the dialog will show:

```
C_LONGINT ($result)
$result:=CS_Register ("")  // display the dialog
```

Note: calling CS_Register with any key (valid or invalid) will not display the dialog.

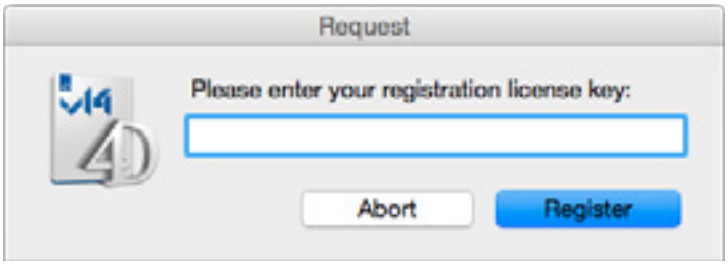
Retrieving the serial/machine information

The Demo mode dialog includes all relevant information (serial or machine ID, see Definitions) to obtain your license, as well as a “Copy” button to put this information into your clipboard or a text file, an “eMail” button to email the information to e-Node’s registration system and a “Register” button to enter your license key once received:

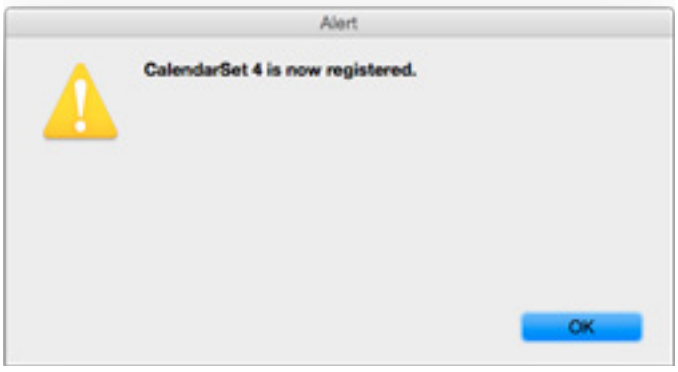


Using the “Register” button

Clicking on this button will display a standard 4D request to enter your registration key:



Paste or drag and drop your registration key and, if correct, the plug-in will be registered for all future uses on this workstation:



Note: if 4D does not activate the **Edit > Paste** menu item click **Abort** and **Register** again, or try drag and drop.

Registering Server licenses

Similarly, server licenses can be registered from the demonstration mode dialog without having to modify your code and use CS_Register (which of course you can do with any license type). In this case, the 4D Licenses folder, serial information or machine ID used will only be the 4D Server information, not the client workstation's.

Server licenses can be registered on any client workstation (remote mode), or on 4D Server itself.

Registering in Remote mode

The server and all workstations can be registered from any single client workstation connected to the server. As in Single user mode, the Demo mode dialog will be displayed on a client workstation when one of the following conditions are met:

- Calling an ITK command other than ***CS_Register*** with a non-empty parameter
- Calling ***CS_Register*** with an empty string

Use the **Copy**, **eMail** and **Register** buttons just as above and your server will be registered for all workstations.

Note: any other workstations previously connected (before registration occurred) will need to re-connect to the server to be functional.

Registering on 4D Server

To directly register the server and all workstations from the server machine itself, you need to display the Demo mode dialog on the server.

Call ***CS_Register*** with an empty string in the **On Server Startup** base method:

```
C_LONGINT ($result)
$result:=CS_Register ("")  // display the dialog
```

Use the **Copy**, **eMail** and **Register** buttons just as above and your server will be registered for all workstations.

Note: the dialog will automatically be dismissed on the server after one minute in order not to block client connections (the server is only available to client workstations once the **On Server Startup** method has completed).

Merged licenses notes

Both methods can be either used with regular or merged servers and client workstations.

- Regular licenses are linked to the 4D Server serial information
- Merged licenses are linked to the 4D Server machine ID

Note: merged licenses will keep working if your 4D Server serial information is modified (upgrading or 4D Partner yearly updates), or if any client workstation hardware is changed.

It will only need to be updated if the 4D Server hardware is changed, or if the plugin itself requires a new key (paid upgrades upon major version changes).

You may want to register your merged server without having to turn off the database to modify the code. We have created a utility database to manage this - it's called Get Serial Info and you can download the appropriate version for your 4D version from the e-Node server.

This is possible using any 4D setup on the server machine (such as a standard developer single user 4D). Keeping your production server alive, open the Get Serial Info database with 4D on the same server machine. Ignore the demonstration mode dialog (if your single user 4D is not registered for the plugin) and wait for the next Alert:

A text file is also saved with the same information.

The last line "Machine ID" is the number that you need to send in order to receive your merged server registration key.

You can also check the machine ID in standalone mode (or on any remote client with the built-client application or in interpreted mode as long as it is running on the same server machine) with AreaList Pro using the following call:

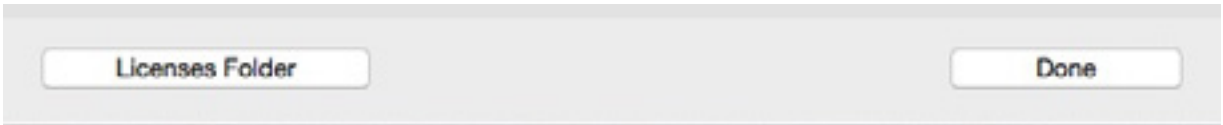
```
C_LONGINT($machineID)
$machineID:= AL_GetAreaLongProperty (0;"mach")
```

Note: you don't need an AreaList Pro license to do this.

Using a text file

Alternately, you can place a plain text file into your 4D Licenses folder.

To open this folder from 4D use the 4D Menu **Help > Update licenses,** then click the **Licenses Folder** button:



The text file **must** be called "CS4.license4Dplugin" and be a plain text type file.

Just paste all your licenses for CalendarSet v4.x, one per line, e.g.:

MyLicense1
MyLicense2
MyLicense3

Any license type can be included into this document, except unlimited single user, OEM and Partner licenses.

Note: the Demo mode dialog **Register** button actually does this: create the text file and include the license key, or add the license key to the existing document if any.

Using CS_Register

1. Open the **On Startup** database method

2. Call the CS_Register function with your registration key - for example:

```
$result:=CS_Register ("YourRegistrationKey")  // result = 0 means registration was successful
```

If you have several licenses for different 4D setups you can call ***CS_Register*** multiple times in a row without further testing. See the Example with multiple calls.

Combining methods

When such a file exists in the Licenses folder CalendarSet will check for valid licenses from this document as a first action before anything else (including checking any ***CS_Register*** command).

If a valid license is included into the “CS4.license4Dplugin” document any calls to ***CS_Register*** will return zero (for “OK”).

Therefore you can mix modes and use the text file (or **Register** button) as well as the command.

Unlimited single user, OEM, temporary and Partner licenses can only be entered through the ***CS_Register*** command.

Online registration

As of version 4.0, CalendarSet provides an automated solution to register itself using an Internet connection.

This feature can be helpful whenever you don’t want to bother your end user with plugin registration, or want to save the time to collect the serial / machine ID, or any other reason when you want the process to be entirely and automatically managed from the client site.

It can also be used for your own development tools, removing the need to modify your 4D code to include or update registration licenses.

Note however that the site must have an open outgoing HTTP Internet connection available.

“Master” keys

The basic principle is that we deliver a non-assigned license key, called “master key”, which you use in your call to CS_Register in your **On Startup** database method. This key will be used to generate valid keys for the plugin and environment, called “final keys”.

One single master key can generate as many final keys as you need, in case you order several licenses of the same kind (regular or merged, single user licenses or server licenses of the same size).

A master key looks like a final key, except that the second part is the plugin code name (same as the license file name) instead of the serial / machine ID, e.g. “123456-CS4-xyz”.

Passing a master key as the first parameter to ***CS_Register*** when the plugin has not been previously registered by any of the methods above will result in a connection attempt to e-Node’s license server as described below.

Process

If the plugin has not been previously registered (through online registration, text file, register button or CS_Register with a final key), and if ***CS_Register*** receives a master key in its first parameter, it will recognize it a such, then:

1. Connect to e-Node's license server.

2. Ask the server if the master key has not been assigned yet (or if the master key is designed to generate several final keys, if there is any unassigned key up to that number).

3. Send the serial information (regular licenses) or the machine ID (merged licenses) to the license server.

4. If an error is detected (such as master key not matching the current setup) return an error to ***CS_Register***.

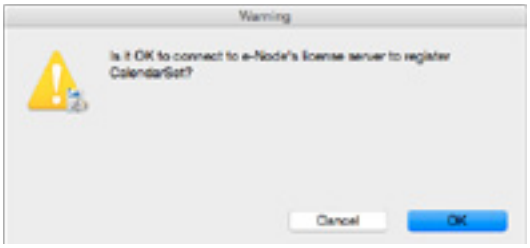
5. If the master key is valid, receive its final key from the license server then register itself (writing into the license file).

Note: if a final key has already been issued for this serial / machine ID using this master key, it is simply resent.

User interface

In addition, CS_Register second parameter allows optional settings regarding the user interface in the online registration process.

Display a confirmation dialog before step 1



Display an alert at step 4

Display an alert at step 5

eMail notification

The third parameter to CS_Register (optional) is the developer email to whom the information will be sent (if this parameter is used and non empty, of course).

The emailed information includes both the final key issued and the IP address from where it was requested (and to where it was sent for registration).

- When a key is issued:

  Title: CS4 license

  Body:

  License 123456-123456789-abcdefgh

  issued to 12.34.56.78

- When a key is resent:

  Title: CS4 license

  Body:

  License 123456-123456789-abcdefgh

  resent to 12.34.56.78

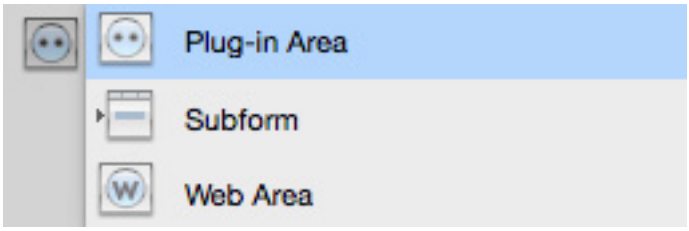
The default mode (master key being passed as the only parameter) is silent: no confirmation, no alert, no email.

Getting Started with CalendarSet

Creating your first CalendarSet Area

It's easy to create your first CalendarSet list area.

1. Create a new form, or open an existing one that you want to add a list to.
2. Choose **Plugin Area** from the Plugin / Subform / Web Area button in the object bar:



3. Your cursor will turn into a crosshair. Draw a box on the form in the size that you want your list to be. This will create a rectangular box named **Plugin Area**.
4. In the Property List window, choose **CalendarSet** from the **Type** popup menu. (If the CalendarSet option is not available, please refer to the installation instructions).
5. Enter a name for your new area in the Variable Name field in the Property List window.
6. Your area will now show the CalendarSet version and copyright information.

The variable name will be used as a parameter for the CalendarSet commands.

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

Advanced Properties or Commands?

You now have a choice: You can configure your list by using the easy-to-use point-and-click interface offered by the Advanced Properties dialog, or you can use the CalendarSet commands to control the list.

You can also use a combination of both methods.

The Advanced Properties dialog doesn't require you to write any code, and is suitable for many projects. However, if you want to have more control over your list, you can use the commands.

You can use both options together: you might use the Advanced Properties dialog to do most of the configuration for a list, and then apply some commands to add some additional programmable control.

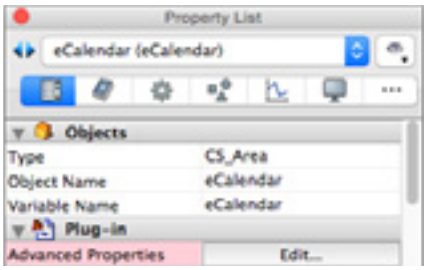
When you do this, the settings specified in the Advanced Properties dialog will be applied when the form is loaded, and then the commands will be applied.

In the following sections we give a brief overview of how to work with both options; they are described in detail in other sections of this manual.

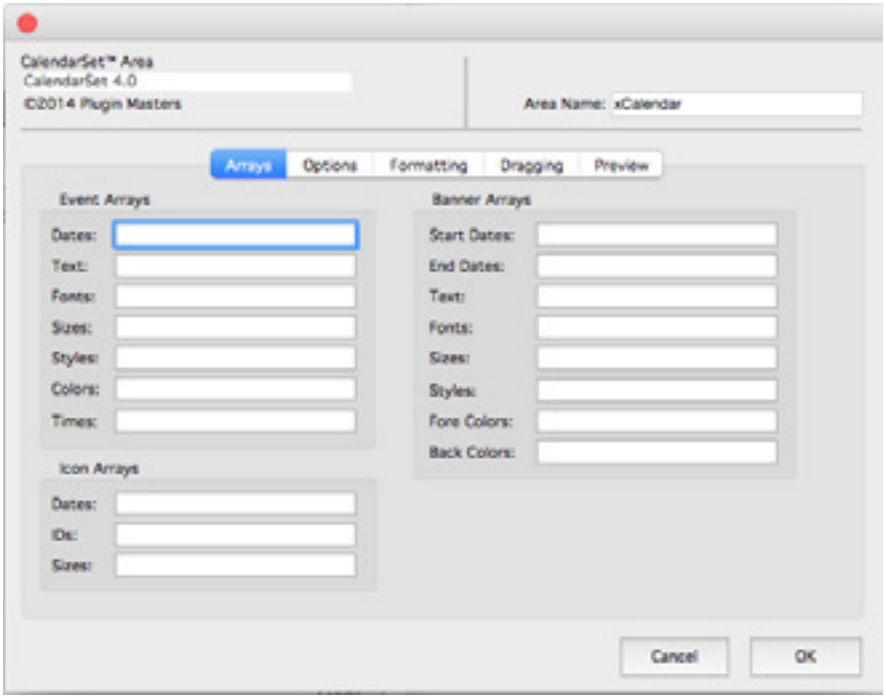
Using the Advanced Properties Dialog

We will have a quick overview of the Advanced Properties Dialog here; you'll find a detailed explanation of it later in this manual.

To invoke the Advanced Properties Dialog, click on the **Edit** button next to **Advanced Properties** in the area's Property List.



The Advanced Properties Dialog opens:



There are a few choices that you **must** make in order for your list to work, and there are lots of other choices that you can make to configure it.

Working with CalendarSet Commands and Functions

You can use the commands and functions to configure every aspect of a CalendarSet area, and to get information about an area.

The commands and functions are grouped into themes: User Action, Configuration, Drag and Drop, etc.

When to use the Commands and Functions

All CalendarSet commands and functions need to be passed a reference to the area on which they will act. Since CalendarSet areas are initialised in the On Load phase of a form, the commands must be called during this phase or afterwards; if you try to call any CalendarSet commands before the form has been loaded, you'll get an error message because 4D does not recognize the area reference.

If you do not issue any CalendarSet commands in the On Load phase, and you haven't chosen any arrays in the Advanced Properties dialog, an empty calendar will be displayed in the CalendarSet area on the form.

You can modify the area by making calls to commands during any other phase or from objects, such as buttons and menus, on the form or in menu bars.

The commands can be used completely independently of the Advanced Properties dialog, or they can work in conjunction with the options you set therein.

For example, you might select the arrays to display in the Advanced Properties dialog and then use some commands to specify different coloring for each event according to some criteria that you specify.

Anatomy of a CalendarSet Command

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. Examples are provided for each of the commands, showing examples of the syntax as well as how the various commands are used together.

Invalid values in command parameters are ignored.

Area reference

The first parameter for most commands is the area reference, depending on the way the plugin area was created:

- variable name (area reference) of the CalendarSet object on the form
- result from the new CS_NewOSArea command (see Offscreen commands)
- result from the **Open external window** 4D command (see CalendarSet in a plugin Window)

This parameter is a long integer, and is required to allow the commands to operate on the correct object.

If an invalid area reference is passed to CalendarSet, the 4D debugger will open in trace (interpreted mode) or an alert is displayed (compiled mode).

Upgrading from Previous versions of CalendarSet

CalendarSet version 4 is compatible with 4D version 13 and above.

To upgrade to CalendarSet version 4, simply install it as described in the Installation section of this manual, replacing your older version.

What's New

Online registration

As of version 4.0, CalendarSet provides an automated solution to register itself using an Internet connection.

This feature can be helpful whenever you don't want to bother your end user with plugin registration, or want to save the time to collect the serial / machine ID, or any other reason when you want the process to be entirely and automatically managed from the client site.

See Online registration in the Installation chapter.

Offscreen mode

CalendarSet v4 allows offscreen areas, which can be set, populated and printed or published on the Web (using CS_GetPicture) without being displayed.

Unicode

CalendarSet supports Unicode for display.

64 bit Server

CalendarSet version 4.0 supports 64 bit server on Windows and MacOS. This feature is only useful for registration or automatic purposes (no user interface), CalendarSet is not meant to be used on a server by a user. However, for example, printing CalendarSet areas will work on a 64 bit server.

Constants

Constants are available for most command parameter values, to help code legibility and maintenance.

New commands

- CS_GetDrgSrcArea returns the source area reference and process when a Drag and Drop has occurred.
- CS_NewOSArea creates an offscreen area.
- CS_DeleteOSArea destroys an offscreen area previously created with ***CS_NewOSArea***.
- CS_SimClick posts a click event in an area, either on screen (same as 4D's **POST CLICK**) or offscreen.
- CS_GetPicture returns a PDF (MacOS) or EMF (Windows) vector picture of the specified area. This feature can be useful for web publishing or offscreen printing, amongst others.

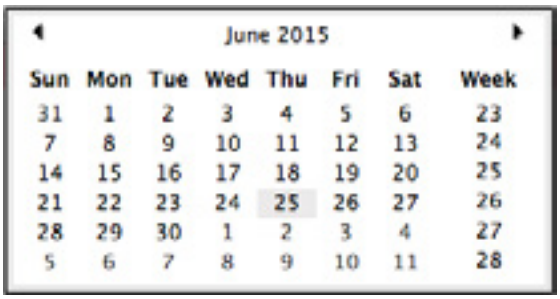
Event times

Time is now supported in events.

- CS_SetArray has a new parameter (**timeArray**), which must contain the name of a longint or time array.
- CS_SetEventOpts has a new parameter: **showEventTimes**. When the column is wide enough, if the **timeArray** was provided and **showEventTimes** is nonzero, events are displayed with the time on the right side.

Alternate “Windows” date popup

An alternate “Windows” look popup date control is available on both platforms when the new **design** parameter to MM_SetOptions is set to 1:



What’s Changed

Drag and Drop

In previous versions, CalendarSet used its own drag and drop manager. It now uses 4D’s drag and drop manager. Drop always occurs in the context of the destination process.

This means that you will need to make a few changes to your drag and drop handling. For more information, see CalendarSet Dragging Commands.

Event Handling

If a drag is initiated, CalendarSet receives a drag event. But this has nothing to do with the actual drop: it could end anywhere (e.g. if the user pressed Esc — no drop). 4D does not inform CalendarSet that the drag was not successful.

When a drop has occurred, an On Drop event is fired. You should therefore use the new CS_GetDrgSrcArea command to find out what was dropped.

Accepting Drops from non-CalendarSet objects

CalendarSet can accept drags from non-CalendarSet objects such as AreaList Pro areas or other 4D objects (in the same process or a different one) and external sources (text selection or document). Furthermore, the drop from external sources can be allowed or disallowed.

Drag and drop to banners

It is now possible to drag and drop to banners.

Native drawing of text

CalendarSet uses CoreText on Mac and GDI+ on Windows.

Only fonts and font faces supported by these technologies can be used in CalendarSet. In particular, GDI+ does not support non-TrueType fonts on some Windows versions.

MM_SetOptions new parameters

MM_SetOptions (area; popIndicator; useDoubleClick; design; colors)

- **popIndicator** is 3-state: 0 = popup arrow(s), 1 = nothing (transparent), 2 = calendar picture
- **useDoubleClick** defaults to 0, which means “close on single click”
- **design** selects the Date Popup variant — identical to AreaList Pro (0 = MacOS, 1 = Windows)
- **colors** is the text specifying the 8 different colors used by the Date Popup — identical to AreaList Pro

TM_SetOptions new parameters

TM_SetOptions (area; popIndicator; useDoubleClick)

- **popIndicator** is 3-state: 0 = popup arrow(s), 1 = nothing (transparent), 2 = time picture
- **useDoubleClick** defaults to 0, which means “close on single click”

Current day background

Current day background is supported (***CS_FontDefaults*** with selector 5, advanced properties).

For compatibility reasons, when the area background color is set (CS_SetCalColor), the current day background color is set to the same value. Set the current day background color after setting the area background color.

Unused days background

Unused days background can be set separately from the background color (third parameter to ***CS_SetCalColor***).

First day of the week

The default value for the first day of week is taken from the system (use -1 to apply the system setting in your call to CS_Options).

Icons

The resource fork model is now completely obsolete. Therefore you must move the icons to be used into the picture library before calling CS_SetIconArray.

Banners

Banners are stacked from the bottom and can be partially visible below the events.

Ellipsis in banners

CalendarSet will automatically truncate data and display the standard ellipsis (…) if the banner width is too small to display the full text. The ellipsis is now displayed in the middle of the banner text instead of at the end.

Obsolete Commands

If you are using any of the following commands, you will need to remove them from your code:

Old Command	How to replace it
CS_DoResize	No longer exists, remove and use 4D options (“Grow”) or commands (**OBJECT MOVE**).
CS_DragMgrAvail	No longer relevant; always true.
CS_GetContRef	Obsolete.
CS_GetDrgArea	Obsolete, use CS_GetDrgSrcArea.
CS_SetSizeOpts	No longer exists, remove and use 4D options (“Grow”) or commands (**OBJECT MOVE**).
Util_SetDate	Deprecated, use 4D’s **Add to date** (!00/00/0000!;$years;$months;$days)

Configuring CalendarSet

Using the Advanced Properties Dialog

CalendarSet 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.

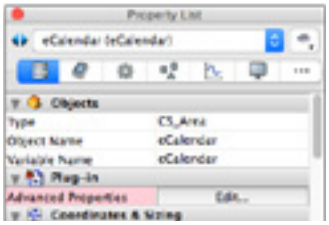
The Advanced Properties dialog lets you specify the names of the arrays to be displayed, almost all options including event / banner selection, color settings, and default styles and color for all CalendarSet text. There is a preview tab to instantly view the options that you've selected.

Once you click the OK button to complete the configuration, the settings will be saved by 4D within the plugin area object on your form. Whenever this form is opened in the user / runtime environments, the settings made here will be applied to your CalendarSet object before the form method or any object methods are executed. Essentially, you are replacing the default settings provided by CalendarSet with new values of your choosing.

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.

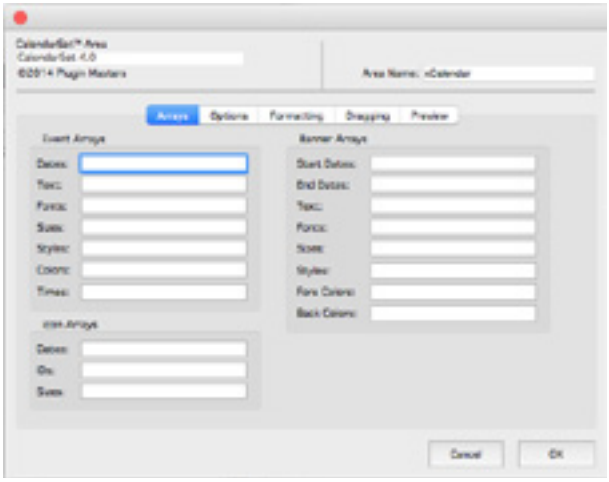


2 Click the Advanced Properties button. The Advanced Properties Dialog will be displayed.

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

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.



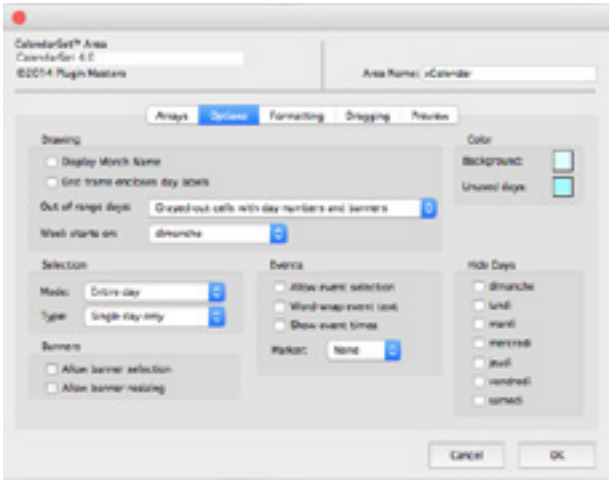
These arrays must be loaded from within your 4D methods. Typically, the arrays will be loaded using the 4D **SELECTION TO ARRAY** command in the On Load event in the CalendarSet object's method.

We recommend that you use the appropriate compiler declarations for all arrays you display, to ensure that you correctly specify the data type of all arrays.

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

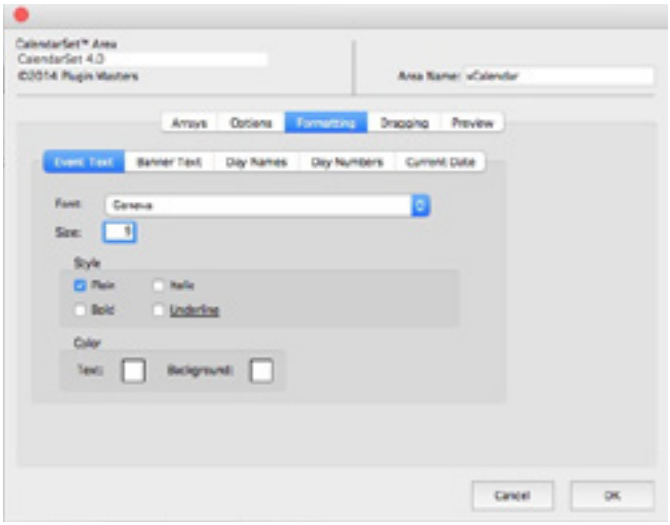
Setting Options

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.



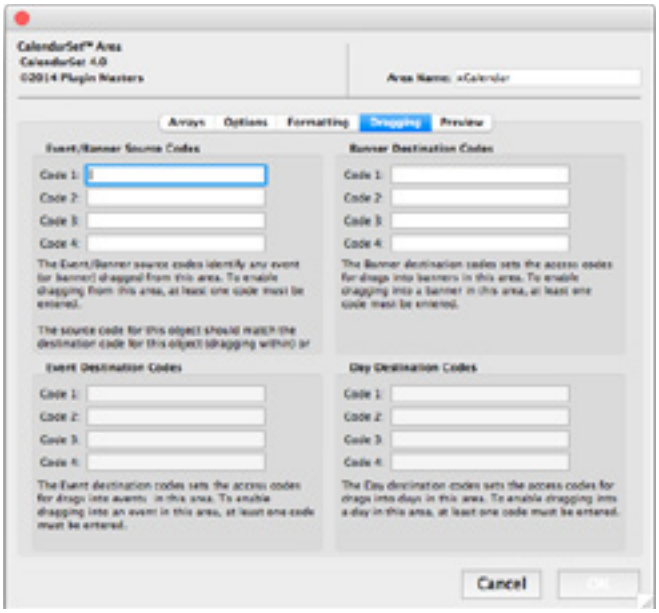
Formatting Options

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.



Dragging

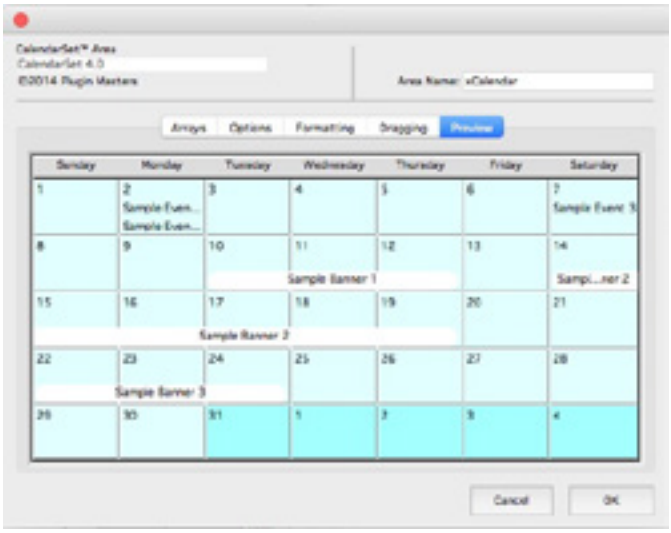
CalendarSet supports drag-and-drop between CalendarSet objects, as well as with AreaList Pro objects, 4D objects or external files. You can configure the dragging behavior using the Dragging pane of the Advanced Properties Dialog. Please read the section Drag and Drop for more information.



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.



Using CalendarSet Commands

CalendarSet commands are used in the On load, On clicked, On Plug in Area and On drop form events.

A CalendarSet object is initialized in the On load phase as the form is about to be displayed.Typically, this initialization will be contained in the object method for the CalendarSet object.

If you use an object method with a CalendarSet object, and you want the object method to execute On load, be sure to check the “On load” checkbox in the object’s Events properties.

Parameters and default values

Most commands use parameters of various types, for which constants are available.

The default value (i.e. if no specific value is set using a CalendarSet command) for numeric parameters is 0 unless otherwise indicated.

Setting Events, Banners and Icons

4D arrays are passed directly to CalendarSet for display, using CS_SetArray for events, CS_SetIconArray for icons, and CS_SetBanrArray 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.

Color table

All color settings used in CalendarSet refer to the same color table.

Each item in the color array allows you to specify either the index into the 4D color palette (positive numbers) or one of the hardcoded (negative) numbers specified in the color table shown below.

Color	Value	Constant
Black	-1	CS_Color_Black
White	-2	CS_Color_White
Red	-3	CS_Color_Red
Green	-4	CS_Color_Green
Blue	-5	CS_Color_Blue
Cyan	-6	CS_Color_Cyan
Magenta	-7	CS_Color_Magenta
Yellow	-8	CS_Color_Yellow
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 Form and Method editors), and count the number of rows down and columns across.

The equation is: **ColorValue = ((RowNumber – 1) x 16) + ColumnNumber**.

Setting Color, Font, Size, and Style Attributes

The default attributes for each type of object on a CalendarSet area are specified using CS_FontDefaults. Colors are set according to the Color table.

The types of objects are day numbers, items, day names, banners, and current day number and background color.

Additionally, you can specify a unique color for a particular item using CS_SetArray, CS_SetBanrArray, and CS_SetDayStyle / CS_SetDayStyleB. 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.

Setting Icons

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

CS_SetIconArray specifies the icons to display, along with the size of the icons.

Displaying Event times

Time is supported in events. If you want to use this feature and display event times:

- The **timeArray** parameter to CS_SetArray contains the name of a longint or time array.
- When the **showEventTimes** parameter to CS_SetEventOpts is set to 1 (or any nonzero value), events are displayed with the time on the right side if the column is wide enough.

Displaying the Month Name

You can display an abbreviation of the month name on the first day of each month, using CS_Options.

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***.

Extending the CalendarSet Frame

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

Specifying the First Day of the Week

The default value for the first day of week is taken from the system. You can change this using ***CS_Options***.

Use -1 to apply the system setting in your call to ***CS_Options***.

Calendar Colors

The colors of the entire calendar may be set from the Color table using CS_SetCalColor.

The background color of unused days (those outside the date range specified for display using CS_SetRange) is also set with ***CS_SetCalColor***.

CS_FontDefaults also sets Current day background (selector value 5). For compatibility reasons, when the area background color is set (***CS_SetCalColor***), the current day background color is set to the same value. Set the current day background color after setting the area 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.

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 discontiguous days to be selected.

Event Selection

You can configure what type of selection to allow using CS_SetEventOpts. The parameter **AllowEventSelect** will consist of two values:

Mode	Value	Constant
No event selection	0	CS_EventSelect_None
Single event selection	1	CS_EventSelect_Single

See Responding to User Actions on a CalendarSet Object for information about how to handle the selection of an event.

Banner Selection and Resizing

Banner selectibility can be set using CS_SetBanrOpts.

You can enable or disable banner resizing by setting **AllowBannerResize** using ***CS_SetBanrOpts***.

When banner resizing is allowed, CalendarSet will automatically update the appropriate elements in the banner arrays that were passed using CS_SetBanrArray. See Responding to User Actions on a CalendarSet Object 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.

This command should be called whenever the CalendarSet object method is executed (in this case, after the resize has occurred). ***CS_GetAction*** returns a value of 5 (CS_Action_BannerResize: user resized a banner).

You can then call CS_GetResizBanr to determine which banner was resized.

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 to 1.

No word wrapping will be performed for banners. CalendarSet will automatically truncate data and display the standard ellipsis (…) if the banner width is too small to display the full text. The ellipsis is displayed in the middle of the banner text.

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 to the desired character followed by a space

Possible marker candidates from the Arial font are "• - Ð Þ —."

Hiding Days

You can hide one or more days in a week using CS_HideDays.

Date format

The date format (MM/DD/YY vs DD/MM/YY) will honor the workstation system settings.

Commands

_CS_Area

_CS_Area is the command used to identify the plugin area when you create an plugin area object on a form.

It is only used in the object definition for a CalendarSet object, and should never be used as a plugin command in a 4D method.

CS_FontDefaults

(AreaRef; Selector; Font; Size; Style; ForeColor; BackColor)

Parameter	Type	Description
→ AreaRef	longint	area reference of the CalendarSet area
→ Selector	integer	attribute identifier
→ Font	text	font name
→ Size	integer	font size
→ Style	integer	font style
→ ForeColor	integer	foreground color
→ BackColor	integer	background color (day names, banners and current day only)

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

Note: invalid values in any characteristic will do nothing.

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

Mode	Value	Constant
Day numbers (cells)	1	CS_Attr_Days
Text that appears in each cell (including time)	2	CS_Attr_Text
Day names that appear at the top of the calendar (headers)	3	CS_Attr_Headers
Banners	4	CS_Attr_Banners
Current date day number and background color	5	CS_Attr_Current

Font — Text. Use this parameter to specify the font for the selected attribute. Pass an empty string ("") to use the default font: Lucida Grande on MacOS and Segoe UI on Windows.

Size — Integer. Use this parameter to set the font size of the selected attribute. Pass 0 to use the default font size for the specified attribute: 9 for all items and both platforms.

Style — Integer. Use this parameter to set the font style of the selected attribute. Pass -1 to use the default font style for the specified attribute: bold for days (CS_Attr_Days), headers (CS_Attr_Headers) and current day (CS_Attr_Current), plain for event text (CS_Attr_Text) and banners (CS_Attr_Banners).

You can use the usual 4D constants (or a combination, e.g. Bold + Italic) to set the styles:

Value	Style
0	Plain
1	**Bold**
2	*Italic*
4	Underline

ForeColor — Integer. Use this parameter to set the foreground color of the selected attribute. Refer to the Color table for possible values. Pass 0 to use the default foreground color for the specified attribute: black for all items.

Note: font name, size, style and foreground color settings for day numbers, day names, and banners can be overridden on an item by item basis using other commands: CS_SetDayStyle, CS_SetDayStyleB, CS_SetArray, and CS_SetBanrArray.

BackColor — Integer. Use this parameter to set the background color of the selected attribute. Refer to the Color table for possible values. Pass 0 to use the default background color for the specified attribute. The defaults are palette index 243 for headers a.k.a. day names (selector 3), -2 (white) for banners (selector 4) and none (use area back color set with CS_SetCalColor) for current day (selector 5).

Note: ***CS_SetBanrArray*** can override the banner background color, too.

Examples

```
// Show day number in Arial 10 point plain, black color
CS_FontDefaults (eCalArea; CS_Attr_Days; "Arial"; 10; 0; 0)
// Show day names in Helvetica 9 point bold, black color
CS_FontDefaults (eCalArea; CS_Attr_Headers; "Helvetica"; 9; 1; 0)
// Set current day background color to light blue (Cyan) leaving the default values for the other attributes
CS_FontDefaults (eCalArea; CS_Attr_Current; ""; 0; -1; 0; CS_Color_Cyan)
```

CS_HideDays

(AreaRef; HideSunday; HideMonday; HideTuesday; HideWednesday; HideThursday; HideFriday; HideSaturday)

Parameter	Type	Description
→ AreaRef	longint	area reference of the CalendarSet area
→ HideSunday	integer	don't display Sunday
→ HideMonday	integer	don't display Monday
→ HideTuesday	integer	don't display Tuesday
→ HideWednesday	integer	don't display Wednesday
→ HideThursday	integer	don't display Thursday
→ HideFriday	integer	don't display Friday
→ HideSaturday	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 1 for any day will hide that day, a value of 0 will show the day. The default for all days is zero (all days are visible by default).

Examples

```
 //Hide Saturday and Sunday
CS_HideDays (eCalArea; 1; 0; 0; 0; 0; 0; 1)

 //Hide Sunday only
CS_HideDays (eCalArea; 1; 0; 0; 0; 0; 0; 0)

 //Show Saturday and Sunday only
CS_HideDays (eCalArea; 0; 1; 1; 1; 1; 1; 0)

 //Show only the current day
CS_SetRange (eCalArea;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 (eCalArea;v1;v2;v3;v4;v5;v6;v7)
```

CS_Options

(AreaRef; DayHighlightMode; DaySelectionType; UnusedInfo; MonthPrefix; FirstDayOfWeek; ExtendFrame)

Parameter	Type	Description
→ AreaRef	longint	area reference of the CalendarSet area
→ DayHighlightMode	integer	type of highlighting to use for day selection
→ DaySelectionType	integer	type of selection allowed
→ UnusedInfo	integer	how to display unused cells
→ MonthPrefix	integer	display the month name for first day of month
→ FirstDayOfWeek	integer	which day starts a week
→ ExtendFrame	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:

Mode	Value	Constant
No Highlight	0	CS_Highlight_None
Highlight entire day's cell	1	CS_Highlight_Cell
Highlight number only	2	CS_Highlight_Number

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

Mode	Value	Constant
Single day only	0	CS_DaySelect_Single
Multiple days selectable (contiguous)	1	CS_DaySelect_Multiple
Discontiguous day selectable	2	CS_DaySelect_Discontiguous

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. There are three possible values:

Mode	Value	Constant
Nothing displayed	0	CS_Unused_Blank
Cell is grayed out	1	CS_Unused_GrayedEmpty
Cell is grayed and day number and banners are displayed (default)	2	CS_Unused_GrayedNumBan

Note: drag and drop onto "out of range" days in the same area will only be allowed if the **UnusedInfo** parameter is set to CS_Unused_GrayedNumBan.

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

Mode	Value	Constant
No month names on first day	0	CS_MonthOnFirstOff
Month names on first day	1	CS_MonthOnFirstOn

FirstDayOfWeek — Integer, -1 to 6. Controls the day of the week on which a week will begin.

Mode	Value	Constant
Use the system setting (default)	-1	CS_FirstDayOS
Sunday	0	CS_FirstDaySun
Monday	1	CS_FirstDayMon
Tuesday	2	CS_FirstDayTue
Wednesday	3	CS_FirstDayWed
Thursday	4	CS_FirstDayThu
Friday	5	CS_FirstDayFri
Saturday	6	CS_FirstDaySat

ExtendFrame — Integer, 0 or 1.

Mode	Value	Constant
CalendarSet will not extend the frame	0	CS_Frame_NotExtended
The frame that is drawn around the CalendarSet area is extended to include the header labels representing the days of the week	1	CS_Frame_Extended

Examples

```
//Highlight number only, multiple cells (contiguous), unused cells grayed, month prefix,
//Sunday start, extend frame
CS_Options (eCalArea;CS_Highlight_Number;CS_DaySelect_Multiple;CS_Unused_GrayedNumBan;\
   CS_MonthOnFirstOn;CS_FirstDaySun;CS_Frame_Extended)
//Highlight entire cell, multiple cells (discontiguous), unused cells blank, no month prefix
//Monday start, don't extend frame
CS_Options (eCalArea;CS_Highlight_Cell;CS_DaySelect_Discontiguous;CS_Unused_Blank;\
   CS_MonthOnFirstOff;CS_FirstDayMon;CS_Frame_NotExtended)
```

CS_Register

(registrationCode; options; email) ➞ result

Parameter	Type	Description
➞ registrationCode	text	Pass the registration key to register your copy of CalendarSet. The key is either linked to the 4D or 4D Server serial number (individual licenses), to the machine ID (merged licenses), to the name of the company/developer (unlimited annual licenses) or to the product (master keys for Online registration)
➞ options	longint	An optional longint combining up to 4 bits: force check, Online registration options
➞ email	text	Online registration option: developer email to notify when a license is issued or resent.
➞ result	longint	0 or error code

CS_Register is used to register the CalendarSet plugin for standalone or server use.

Please see the License Types section for detailed information about the licensing options available for CalendarSet.

Multiple calls to ***CS_Register*** are allowed. The plugin will be activated if at least one valid key is used, and all subsequent calls to ***CS_Register*** will return 0, unless the force check bit is set to true in the **options** parameter.

registrationCode — You must call ***CS_Register*** with a valid registration key, otherwise CalendarSet will operate in demonstration mode - it will cease to function after 20 minutes. In case a master key is used the plugin will attempt a connection to e-Node's license server for Online registration.

options — Optional. This parameter combines up to 4 bits as described below. The default mode (**registrationCode** being a passed as the only parameter) is silent: no force check, no confirmation, no alert, no email.

Bit number	Description
0	Force check: if this bit is is on (true), **registrationCode** is tested regardless of current registration state. If the plugin was not previously registered and the result is 0, it is registered the same way as if the bit was off (or the whole **options** parameter omitted) If the plugin was previously successfully registered, a registration error will be returned in result in case **registrationCode** is invalid, but the plugin will remain registered.
1	Online registration option: confirm connection "Is it OK to connect to e-Node's license server to register CalendarSet?"
2	Online registration option: display alert if registration error
3	Online registration option: display alert if registered

email — Optional. The developer email address where to send Online registration information.

result — 0 or error code:

Result code	Description
0	OK
1	Beta license has expired
2	Invalid license
3	The license has expired
4	The OEM license has expired
5	The maximum number of users has been exceeded
6	The license is for a different environment (e.g. the licence is for a single-user version, but you are using it with 4D Server)
7	The license is linked to a different 4D license
8	Invalid merged license
9	Only serial/ID licenses are allowed in text license files (includes Register button and Online registration)
10	Unauthorized master key (Online registration)
11	Can't connect to e-Node's license server to perform Online registration
12	No Online registration license available for this master key (unknown or all used)

When ***CS_Register*** is called with an empty string, the license dialog will be displayed if CalendarSet is not registered and the dialog was not yet displayed. This allows you to show the registration dialog to your users without effectively calling a CalendarSet command or displaying a CalendarSet area.

Note: alternately to ***CS_Register***, you can place a plain text file into your 4D Licenses folder or use the Demo mode dialog "Register" button. This is only valid for non-unlimited licenses.

Basic example

```
C_LONGINT ($result)
$result:=CS_Register ("YourRegistrationKey")
Case of
   :($result=2)
      ALERT ("The CalendarSet licence is invalid.")
   :($result=3)
      ALERT ("The CalendarSet licence has expired.")
   etc.
End case
```

Example with multiple calls

```
C_LONGINT ($result)  //ignored in this case
$result:=CS_Register ("Registration key one")
$result:=CS_Register ("Registration key two")
$result:=CS_Register ("Registration key three")
```

etc.

```
If ($result#0)  //registration failed on all keys
   ALERT ("CalendarSet could not be registered.")
End if
```

Force check example

In this example we assume that only "Registration key two" is valid, but you want to check the other keys status.

```
C_LONGINT ($result)
$result:=CS_Register ("Registration key one";1)  //invalid, will return an error, the plugin isn't registered
$result:=CS_Register ("Registration key two";1)  //valid, will return 0, the plugin is registered
$result:=CS_Register ("Registration key three";1)  //invalid, will return an error, the plugin is still registered
```

Online registration examples

Confirm connection, alert if successful, alert if failed, send email notification to developer@4dchampions.com:

```
C_LONGINT ($result)
$result:=CS_Register ("Master key";0 ?+1 ?+2 ?+3;" developer@4dchampions.com")
```

Silent connection, alert if successful, alert if failed, no email notification:

```
C_LONGINT ($result)
$result:=CS_Register ("Master key";0 ?+2 ?+3)
```

CS_SetArray

(AreaRef;DateArray;TextArray; FontArray; SizeArray; StyleArray; ColorArray; TimeArray)

Parameter	Type	Description
→ AreaRef	longint	area reference of the CalendarSet area
→ DateArray	text	dates of each item
→ TextArray	text	text of each item to display (name of text array)
→ FontArray	text	font to use for each item (name of text array)
→ SizeArray	text	size of each item (name of numeric array)
→ StyleArray	text	style of each item (name of numeric array)
→ ColorArray	text	color of each item (name of numeric array)
→ TimeArray	text	time of each item (name of long integer or time array)

CS_SetArray is used to specify the arrays that are 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 to force CalendarSet to show the changes.

Note: modifying the event arrays doesn't require reissuing ***CS_SetArray***, but you must call ***Area_Refresh***.

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

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

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

TextArray — Text (name of a text array). Used to define the text of the events to be displayed.

Note: **DateArray** and **TextArray** are required (otherwise no events will be displayed).

FontArray — Text (name of a text array). The referenced array contains the fonts to display each corresponding event. Use an empty text array to use the default font value for all events.

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

StyleArray — Text (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. Use an empty numeric array to use the default style value for all events.

ColorArray — Text (name of a numeric array). Each item in the color array allows you to specify the color from the Color table. Use an empty numeric array to use the default color value for all events.

TimeArray — Text (name of a longint or time array). Used to define the times of the events to be displayed.

You can omit the styling arrays using empty strings.

For example, assigning "" as **ColorArray** will have the effect of ignoring **FontArray**, **SizeArray** and **StyleArray**, not only **ColorArray**.

You have to provide an array name, but this array can be an empty **ARRAY INTEGER** (or correctly sized but containing zeros).

Notes:

1. When an empty string is provided as the array name (or the name does not resolve to an array of the proper type), **all** styling arrays are ignored.
2. When the name of an empty array is provided, the default for that styling property is used for all (missing) elements.
3. When the array has less elements than needed, default for that styling is used for missing elements.
4. To use default for a given element, use:
   - empty text for **FontArray**
   - -1 for **StyleArray**
   - 0 for all other styling arrays

Example

```
 //Display all items for this month from the appointments file
C_DATE(vDay1ThisMth;vDay1NextMth)
vDay1ThisMth:= Add to date(!00/00/0000!;Year of(Current date);Month of(Current date);1)
vDay1NextMth:= Add to date(vDay1ThisMth;0;1;0)
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, color and time.
 //this makes loading the data and displaying it very easy to accomplish
SELECTION TO ARRAY([Appts]Appt Date;aDates;[Appts]Item;aItems;[Appts]Item Font;\
   aFonts;[Appts]Size;aSizes;[Appts]Styles;aStyles;[Appts]Color;aColors;[Appts]Time;aTimes)
 //and display the data in the CalendarSet plugin area named "eMonth"
CS_SetArray (eMonth;"aDates";"aItems";"aFonts";"aSizes";"aStyles";"aColors";"aTimes")
```

CS_SetBanrArray

(AreaRef; StartDateArray; EndDateArray; TextArray; FontArray; SizeArray; StyleArray; ForeColorArray; BackColorArray)

Parameter	Type	Description
→ AreaRef	longint	area reference of the CalendarSet area
→ StartDateArray	text	start dates for each banner
→ EndDateArray	text	end dates for each banner
→ TextArray	text	text for each banner
→ FontArray	text	font for each banner
→ SizeArray	text	size for each banner
→ StyleArray	text	style for each banner
→ ForeColorArray	text	foreground color for each banner
→ BackColorArray	text	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.

The arrays are passed in quotes, which is why the parameter type is text. The size, style, foreground color, and background color arrays are of type integer.

Note: modifying the banner arrays doesn't require reissuing ***CS_SetBanrArray***, but you must call Area_Refresh.

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

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

TextArray — Text (name of a text array). Used to define the text of the banners to be displayed.

Note: **StartDateArray**, **EndDateArray** and **TextArray** are required (otherwise no banners will be displayed).

FontArray — Text (name of a text array). The referenced array contains the fonts to display each corresponding banner.

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

StyleArray — Text (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 — Text (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 the color from the Color table.

BackColorArray — Text (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 the color from the Color table.

See CS_SetArray regarding default values and omitting styling arrays.

Example

```
CS_SetBanrArray (eMonth;"aStDates";"aEnDates";"aItemTexts";"aFonts";"aSizes";"aStyles";\
    "aForeColors";"aBackColors")
```

CS_SetBanrOpts

(AreaRef; AllowBannerSelect; AllowBannerResize)

Parameter	Type	Description
→ AreaRef	longint	area reference of the CalendarSet area
→ AllowBannerSelect	integer	allow selection of banners
→ AllowBannerResize	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. You can detect that a banner has been resized using CS_GetResizBanr.

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

Mode	Value	Constant
No bannner selection allowed	0	CS_Banner_NoSelect
Single banner select	1	CS_Banner_Select

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

Mode	Value	Constant
Banner resizing not allowed	0	CS_Banner_NoResize
The user will be able to resize banners by clicking on either of the banner's flags and dragging to a new date	1	CS_Banner_Resize

Default values

- If ***CS_SetBanrOpts*** is not used, both **AllowBannerSelect** and **AllowBannerResize** default to 1.
- If ***CS_SetBanrOpts*** is used, both default to 0 (if the parameter is not used).

Examples

```
//Allow banner selection, allow banner resizing
CS_SetBanrOpts(eCalArea;CS_Banner_Select;CS_Banner_Resize)
//No banner selection, no banner resizing
CS_SetBanrOpts(eCalArea;CS_Banner_NoSelect;CS_Banner_NoResize)
```

CS_SetCalColor

(AreaRef; CalForeColor; CalBackColor)

Parameter	Type	Description
→ AreaRef	longint	area reference of the CalendarSet area
→ CalForeColor	integer	color of calendar unused days
→ CalBackColor	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) 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. The default is palette index 193.

CalBackColor — Integer. Refer to the Color table. The default is palette index 195.

Example

```
//Unused days color blue, background color very light gray
CS_SetCalColor(eCalArea; CS_Color_Blue;241)
```

CS_SetDayStyle

(AreaRef; TargetDate; DayFont; DaySize; DayStyle; DayColor; ClearOthers)

Parameter	Type	Description
→ AreaRef	longint	area reference of the CalendarSet area
→ TargetDate	date	date to change the font info
→ DayFont	text	font name for the day number
→ DaySize	integer	font size for the day number
→ DayStyle	integer	font style for the day number
→ DayColor	integer	font color for the day number
→ ClearOthers	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 — Text. 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 Color table possible color values.

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 using special font info.

Example

```
// Display the current date in Times 12 point bold, black color
// set all other days to the default style
CS_SetDayStyle (eCalArea; Current date; "Times"; 12; 1; 0; 1)
```

CS_SetDayStyleB

(AreaRef; DateArray; DayFont; DaySize; DayFace; DayColor; ClearOthers)

Parameter	Type	Description
→ AreaRef	longint	area reference of the CalendarSet area
→ DateArray	date array	dates to change the font info
→ DayFont	text	font name for the day number
→ DaySize	integer	font size for the day number
→ DayStyle	integer	font style for the day number
→ DayColor	integer	font color for the day number
→ ClearOthers	integer	clear custom info for all other days

CS_SetDayStyleB is identical to CS_SetDayStyle 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 (eCalArea; aDates; "Palatino"; 10; 3; 0; 0)
```

CS_SetEventOpts

(AreaRef; AllowEventSelect; ShowWordWrap; EventMarker; ShowEventTime)

Parameter	Type	Description
→ AreaRef	longint	area reference of the CalendarSet area
→ AllowEventSelect	integer	allow selection of events
→ ShowWordWrap	integer	word-wrap event text
→ EventMarker	integer	character to mark each event
→ ShowEventTime	integer	show times next to events

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.

Mode	Value	Constant
No event selection	0	CS_EventSelect_None
Single event selection	1	CS_EventSelect_Single

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.

Mode	Value	Constant
No word wrapping will be performed	0	CS_WordWrap_Off
Word wrapping will be performed on all events	1	CS_WordWrap_On

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).

Mode	Value	Constant
No marker	0	CS_Marker_None
Bullet	1	CS_Marker_Bullet
Dash	2	CS_Marker_Dash

ShowEventTime — Integer. 0 or 1. This parameter is used to display the times associated to the events. This feature requires that a **TimeArray** has been set using the CS_SetArray command.

Mode	Value	Constant
Don't show event times (even if a time array has been set)	0	CS_EventTime_Hide
Show event times (provided that a time array has been set)	1	CS_EventTime_Show

Default values

- If ***CS_SetEventOpts*** is not used, all parameters following **AreaRef** default to 0.
- If ***CS_SetEventOpts*** is used, all default to 0 (if the parameter is not used).

Examples

```
//Allow event selection, word-wrap event text, mark events with bullet, don't show event times
CS_SetEventOpts(eCalArea;CS_EventSelect_Single;CS_WordWrap_On;CS_Marker_Bullet;CS_EventTime_Hide)
//No event selection, word-wrap event text, mark events with dash, show event times
CS_SetEventOpts(eCalArea;CS_EventSelect_None;CS_WordWrap_On;CS_Marker_Dash;CS_EventTime_Show)
```

CS_SetEvtSelect

(AreaRef; Type; EventIndex)

Parameter	Type	Description
➞ AreaRef	longint	area reference of the CalendarSet area
➞ Type	integer	event or banner
➞ EventIndex	integer	array element corresponding to item being selected

CS_SetEvtSelect is used to procedurally select events or banners.

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

Type	Value	Constant
Event	1	CS_Type_Event
Banner	2	CS_Type_Banner

EventIndex — Integer. Index within the array that was passed to CalendarSet via either CS_SetArray or CS_SetBanrArray.

Use CS_GetEvtSelect to determine what event or banner was selected by the user.

Examples

```
//Set the first event to be selected
CS_SetEvtSelect(eCalArea;CS_Type_Event;1)
//Set the third event to be selected
CS_SetEvtSelect(eCalArea;CS_Type_Event;3)
//Set the fourth banner to be selected
CS_SetEvtSelect(eCalArea;CS_Type_Banner;4)
```

CS_SetIconArray

(AreaRef; DateArray; IconArray; SizeArray)

Parameter	Type	Description
→ AreaRef	longint	area reference of the CalendarSet area
→ DateArray	text	date for display of each icon
→ IconArray	text	picture library numbers for each icon
→ SizeArray	text	size for each icon

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

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

Note: calling ***CS_SetIconArray*** after making changes to the icon arrays is more effective than Area_Refresh.

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

IconArray — Text. This parameter is the name of an integer array and specifies the 4D picture library IDs of the icons to display. The icons specified in **IconArray** must be stored in the picture library. There can be, at most, one icon per day in a calendar.

SizeArray — Text. This parameter is the name of an integer array and specifies the length in pixels of the side of the frame in which each icon will be drawn (scaled proportionally, centered). The following values are allowed:

Mode	Value	Constant
Best Size — CalendarSet will draw the size that fits best depending on the available space in the day cell	0	CS_Icon_Best
Small Icons (16 x 16)	16	CS_Icon_Small
Large Icons (32 x 32)	32	CS_Icon_Large

Example

```
SELECTION TO ARRAY([Dates]Appt Date;aIconDates;[Dates]Icon ID;aIconIDs;[Dates]Icon Size;aIconSizes)
 // The icon info is in the arrays aIconDates, aIconIDs, and aIconSizes
CS_SetIconArray(eCalArea;"aIconDates";"aIconIDs";"aIconSizes")
 //Add icon #2118 from the 4D picture library to the date 4/12/2015 within a 32 x 32 pixels square
$Position:=Find in array(aIconDates;!04/12/2015!)  //see if the date is already in the appointment dates list
If ($Position=-1)  //not there yet
   APPEND TO ARRAY(aIconDates;!04/12/2015!)  //let's add it
   $Position:=Size of array(aIconDates)
   ARRAY INTEGER(aIconIDs;$Position)
   ARRAY INTEGER(aIconSizes;$Position)
End if
aIconIDs{$Position}:=2118
aIconSizes{$Position}:=CS_Icon_Large
Area_Refresh (eCalArea)
```

CS_SetRange

(AreaRef;StartDate;EndDate)

Parameter	Type	Description
➞ AreaRef	longint	area reference of the CalendarSet area
⇆ StartDate	date	the first date to be displayed in the calendar
⇆ EndDate	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.

Note: **StartDate** and **EndDate** are input/output parameters as explained below.

StartDate — Date. This value will set the first date for the Calendar. Pass a variable to retrieve the current value.

Note: an invalid value to **StartDate** will set it to the first day of the current month.

EndDate — Date. This value will set the last date for the Calendar. Pass a variable to retrieve the current value.

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

Note: if **StartDate** is after **EndDate** the values are exchanged.

The current selection of cells may be lost when this command is called: the selected days will be cleared when **StartDate** is modified or the number of weeks is changed (calling ***CS_SetRange*** with current values for **StartDate** and **EndDate** will not clear the selection).

To maintain the selection, use CS_GetSelect prior to calling ***CS_SetRange***, and CS_SetSelect afterward.

Examples

- Set the calendar to display the month of March, 2015 (MM/DD/YY format):
  CS_SetRange (eCalArea;!03/01/2015!;!00/00/00!)
- Get the starting and ending dates of the currently displayed calendar:
  CS_SetRange (eCalArea;vStartDate;vEndDate)

CS_SetSelect

(AreaRef;StartDate;EndDate;Contiguous;DaysArray)

Parameter	Type	Description
→ AreaRef	longint	area reference of the CalendarSet area
→ StartDate	date	the start date of the selection
→ EndDate	date	the end date of the selection
→ Contiguous	integer	1 if selection is contiguous, 0 if not
→ DaysArray	date array	array of currently selected items

CS_SetSelect sets the current selection of the calendar. This is the opposite action of CS_GetSelect.

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.

Mode	Value	Constant
The current selection is set to be each of the days that are specified in **DaysArray**	0	CS_Sel_Discontiguous
The current selection is set to the range **StartDate** to **EndDate**	1	CS_Sel_Contiguous

DaysArray — Date array. This parameter is used to pass a list of discontiguous 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 (eCalArea;vStart;vEnd;1;aDate)
 // Highlight the first three days of March 2015 (MM/DD/YY format)
CS_SetSelect(eCalArea;!03/01/15!;!03/03/15!;CS_Sel_Contiguous;aDate)
 // Highlight the 1st and 15th of May 2015 (MM/DD/YY format):
ARRAY DATE(aDate;2)
aDate{1}:=!05/01/15!
aDate{2}:=!05/15/15!
CS_SetSelect(eCalArea;!00/00/00!;!00/00/00!;CS_Sel_Discontiguous;aDate)
```

User Action Commands

Responding to User Actions on a CalendarSet Object

User interaction with a CalendarSet object is handled in the On load, On plug in area, On Clicked and On drop form events. Once again, the CalendarSet object method or form method, or callback will contain the procedures to detect user actions, such as selection, double-clicks, etc.

You may also use certain CalendarSet commands in other 4D methods, such as a query button object method which changes the arrays currently displayed in a CalendarSet object.

In such a method, if you modify the banner/event arrays in any way (other than calling CS_SetArray/CS_SetBanrArray), you'll need to use Area_Refresh to update the CalendarSet object.

Note: CalendarSet will update automatically when area settings are modified.

The user's action is communicated to you using CS_GetAction whenever the CalendarSet object method or form method, or callback method (or for an plugin window, the callback method only) 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 to determine the event that is selected. CS_LastClick 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, which is similar in syntax to CS_GetEvtSelect.

Note: if you are referencing the area through a pointer, make sure that you use an expression to make sure that CalendarSet does not receive 0 due to 4D limitations:

Don't use

```
CS_GetAction ($srcObject->;vAction)  // To obtain the action of the object
```

use

```
CS_GetAction (($srcObject->);vAction)
```

or

```
CS_GetAction ($srcObject->+0;vAction)
```

Commands

CS_GetAction

(AreaRef;ActionCode)

Parameter	Type	Description
→ AreaRef	longint	area reference of the CalendarSet area
← ActionCode	integer	type of user action

This routine is called from the CalendarSet object method (or for a CalendarSet plugin window, from the **MethodToExecute**) See CS_SetCallback in order to determine the last CalendarSet action. Please read the section Responding to User Actions on a CalendarSet Object for more information.

ActionCode — Integer, 1 to 8. This parameter is returned with a value indicating what action the user made on the CalendarSet object.

Possible values are:

Action	Value	Constant
User click on day (no event selected)	1	CS_Action_ClickOnDay
User closed the window (valid only for a CalendarSet plugin window)	2	CS_Action_WindowClosed
User clicked on an event or banner	3	CS_Action_ClickOnEventBanner
User dragged an event or banner (deprecated)	4	
User resized a banner	5	CS_Action_BannerResize
User resized the window (obsolete)	6	
User dropped something onto the area	7	CS_Action_Drop
Dragging a non CalendarSet/AreaList Pro object over the area	8	CS_Action_DragOver

Example

```
// eCalendar object method, detect a click on a day, an event or a banner and act accordingly
C_LONGINT(vAction;vEvtType;vIndex)
CS_GetAction (eCalendar;vAction)
Case of
   :(vAction=CS_Action_ClickOnDay)  // clicked on a day
      ShowDay
   :(vAction=CS_Action_ClickOnEventBanner)  // clicked on an event or banner
      CS_GetEvtSelect (eCalendar;vEvtType;vIndex)
      Case of
         : (vEvtType=CS_Type_Event)  // event
            ModifyDay (vIndex)
         : (vEvtType=CS_Type_Banner)  // banner
            ModifyBanner (vIndex)
      End case
End case
```

CS_GetEvtSelect

(AreaRef; Type; EventIndex)

Parameter	Type	Description
→ AreaRef	longint	area reference of the CalendarSet area
← Type	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 prior to this command.

Only one event can be selected at a time (requires CS_SetEventOpts to be called with the **AllowEventSelect** parameter set to CS_EventSelect_Single).

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

Type	Value	Constant
Event	1	CS_Type_Event
Banner	2	CS_Type_Banner

EventIndex — Integer. This value is an index within the array that was passed to CalendarSet via either CS_SetArray or CS_SetBanrArray.

Use CS_SetEvtSelect to procedurally select an event or banner.

Example

```
C_LONGINT(vAction;vType;vIndex)
If(Form event=On Clicked)
   CS_GetAction(eCal;vAction)
   If(vAction=CS_Action_ClickOnEventBanner)  //did the user select an event or banner?
      CS_GetEvtSelect(eCal;vType;vIndex)
      Case of
         :(vType=CS_Type_Event)  //event
            ALERT("Event "+aEvent{vIndex}+" was selected.")
         :(vType=CS_Type_Banner)  //banner
            ALERT("Banner "+aBanner{vIndex}+" was selected.")
      End case
   End if  //CS_Action_ClickOnEventBanner
End if  //On Clicked
```

CS_GetResizBanr

(AreaRef; BannerIndex)

Parameter	Type	Description
➞ AreaRef	longint	area reference of the CalendarSet area
⬅ 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 object method (or callback method for an plugin window) after first using CS_GetAction and retrieving an action code of 5 (CS_Action_BannerResize), indicating the user has resized a banner.

BannerIndex — Integer. Index of the resized banner within the arrays passed to CalendarSet using CS_SetBanrArray.

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

Example

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

CS_GetSelect

(AreaRef;StartDate;EndDate;Contiguous;DaysArray)

Parameter	Type	Description
→ AreaRef	longint	area reference of the CalendarSet area
← StartDate	date	the start date of the selection
← EndDate	date	the end date of the selection
← Contiguous	integer	1 if selected dates are adjacent to each other
← DaysArray	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 (1) or not (0) all of the cells that are selected are contiguous (located adjacent to each other).

Mode	Value
The current selection is made of the days that are specified in **DaysArray**	0
The current selection covers the range **StartDate** to **EndDate**	1

DaysArray — Date array (optional). If passed, this array will be filled with the dates of each of the selected items in the calendar, whether the selection is contiguous or not.

Use CS_SetSelect 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)
CS_GetSelect (eCalendar;vStart;vStop;vContiguous;aDates)  // get the selected days
// now vContiguous = 1 if the selection is contiguous
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_GetSelItems

(AreaRef; ItemType; SelectedItems)

Parameter	Type	Description
→ AreaRef	longint	area reference of the CalendarSet area
→ ItemType	integer	kind of item to return
← SelectedItems	Integer array	index of all items belonging to selected days

CS_GetSelItems will return each of the array items (events or icons) belonging to the selected days in the calendar.

You will usually use this command after first using CS_GetAction to determine that the user has made a selection.

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

Type	Value	Constant
Events	1	CS_Sel_Events
Icons	2	CS_Sel_Icons

SelectedItems — Integer array. This parameter contains the index of each item in the array that corresponds to a selected day in the calendar. See FS_ArrayIntrsct for an example of this command.

Example

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

If(Form event=On Clicked)
   CS_GetAction(eCal;vAction)

   If(vAction=CS_Action_ClickOnDay)  // did the user select a day or days?
    // get the events on this day or these days
      CS_GetSelItems(eCal;CS_Sel_Events;aIndex)

      For($i;1;Size of array(aIndex))
       // do something with each event's arrays
       // for this example we'll put a bullet character at the beginning of each event's text
         aEvent{aIndex{$i}}:="• "+aEvent{aIndex{$i}}
      End for
      Area_Refresh (eCal)  // notify CalendarSet

   End if  // CS_Action_ClickOnDay
End if  // On Clicked
```

CS_LastClick

(AreaRef; DateClicked; WasDouble; Modifiers)

Parameter	Type	Description
→ AreaRef	longint	area reference of the CalendarSet area
← DateClicked	date	date that was clicked by user
← WasDouble	Integer	was event a double-click?
← Modifiers	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 or an event is selected.

WasDouble — Integer. This parameter indicates if the user has double-clicked (1) or single-clicked (0).

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

Modifier Key	Value	Constant (4D)
None	0	
Ctrl / command (Windows = Ctrl key, MacOS = Command key)	256	Command key mask
Shift	512	Shift key mask
Caps Lock	1024	Caps lock key mask
Option (Windows = Alt key, Mac OS = Option key)	2048	Option key mask
Control (MacOS only)	4096	Control key mask

Example

```
//eCalendar object method, detect a double-click on a day
//If command key depressed, add a banner
//Otherwise, add an event
Case of
   :(Form event=On Clicked)
      CS_GetAction (eCalendar;vAction)
      If(vAction=CS_Action_ClickOnDay)  //clicked on a day
         CS_LastClick (eCalendar;vDate;vDC;vMod)
         If(vDC=1)  //was it a double-click?
            If(vMod & Command key mask)#0)  //command key?
               ADD CAL BANNER (vDate)
            Else
               ADD CAL EVT (vDate)
            End if  //command key
         End if  //double-click
      End if  //CS_Action_ClickOnDay
End case
```

CS_SetCallback

(AreaRef; MethodToExecute)

Parameter	Type	Description
→ AreaRef	longint	area reference of the CalendarSet area
→ MethodToExecute	text	method to execute when calendar is clicked

CS_SetCallback allows you to associate a method with a calendar area.

The method will be executed whenever the calendar is clicked on or other actions are performed by the user, such as closing the window, resizing a banner, or drag and drop. This command is normally used to detect and act on a user action on a CalendarSet plugin area or window.

MethodToExecute — Text. When CalendarSet detects a user action on a calendar area, the specified method will be executed. CalendarSet will pass two parameters to the method.

The first parameter, **AreaRef**, is received by the callback method in $1, and is of type long integer.

The second parameter, received 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, are shown below.

Action	Value	Constant
User click on day (no event selected)	1	CS_Action_ClickOnDay
User closed the window (valid only for a CalendarSet plugin window)	2	CS_Action_WindowClosed
User clicked on an event or banner	3	CS_Action_ClickOnEventBanner
User dragged an event or banner (deprecated)	4	
User resized a banner	5	CS_Action_BannerResize
User resized the window (obsolete)	6	
User dropped something onto the area	7	CS_Action_Drop
Dragging a non CalendarSet/AreaList Pro object over the area	8	CS_Action_DragOver

The drag action (value 4) is deprecated: drag and drop should be handled entirely when the drop occurs, with the CS_Action_Drop action. See Drag and Drop.

With CS_Action_DragOver only (when the dragged object is an external source), the callback method can return a $0 result, to selectively accept or refuse the drop. See Using the callback method and Accepting a Drag from a non-CalendarSet Object.

When testing the value of the second parameter, a Case statement is recommended.

Example

```
// Open a CalendarSet object in an plugin window.
// Configure the calendar for single day selection.
// Highlight the entire day, show info in unused days, display month prefix.
// Show the Current date in Times 14 point bold.
// Set the click method to be "HandleCalClick".
vCalWindow:= Open external window(50; 50; 400; 400; 4; "Calendar Window"; "_CS_Area")
      // open the window with CalendarSet displayed
CS_Options  (vCalWindow;CS_Highlight_Cell;CS_DaySelect_Single;CS_Unused_GrayedNumBan;CS_MonthOnFirstOn)
CS_SetDayStyle (vCalWindow;Current date;"Times";14;1)
CS_SetCallback(vCalWindow;"HandleCalClick")
```

```
 //Project method HandleCalClick
C_LONGINT($1;$CalArea)  //CalendarSet object reference
C_INTEGER($2;$Action)  //type of user action
$CalArea:=$1  //reassign received parameters for better readability
$Action:=$2
Case of
   :($Action=CS_Action_ClickOnDay)  //user click on a day
      CS_GetSelect ($CalArea;vDateSt;vDateEn;vContig;aDays)
      ModDayInfo ($CalArea;vDateSt)  //add/modify the info for that day
   :($Action=CS_Action_WindowClosed)  //user click in plugin window close box or the hosting form is closed
      SaveCalendar  //save any changes
   :($Action=CS_Action_ClickOnEventBanner)  //clicked on an event or banner
   :($Action=CS_Action_BannerResize)  //resized a banner
   :($Action=CS_Action_Drop)  //dropped an event or banner (or a non-CalendarSet object)
    //handle the drag and drop here
   :($Action=CS_Action_DragOver)  //dragging a non CalendarSet/AreaList Pro object over the area
      $0:=1  //allow drop
End case
```

CalendarSet Drag and Drop Commands

Drag and Drop

Drag and drop of events and banners onto days, events and banners is available from and to CalendarSet areas.

Overview

When an item is clicked and dragged, the cursor will change.

If the drop is allowed, the cursor will have a plus symbol attached to it when it hovers over the “drop” area: 

If the drop is not allowed by the destination object, you’ll see a “no entry” sign instead: 

Dragging

You can allow events and banners to be dragged from a CalendarSet area and dropped to various destinations:

- a day, an event or a banner in the same CalendarSet area
- a day, an event or a banner in another CalendarSet area
- an AreaList Pro area
- any 4D droppable object
- another application that accepts text

Dropping

You can also allow dropping onto days, events and / or banners in a CalendarSet area from various sources:

- events and banners in the same CalendarSet area
- events and banners in another CalendarSet area
- rows, columns or cells in an AreaList Pro area
- any 4D draggable object
- text or other contents displayed in another application window
- a document in a MacOS Finder or Windows Explorer window

Controlling the Drag and Drop

You can control each CalendarSet area's draggability and droppability:

- if the area can be dragged from
- if the area can be dropped to
- which item types (days, events and/or banners) can be dropped to
- which CalendarSet or AreaList Pro areas can be the source or destination
- if external sources (non-CalendarSet/AreaList Pro objects) are allowed
- whether an attempted a drop on the area is accepted or rejected
- what happens after a drop

Configuring Drag and Drop

You must configure CalendarSet to allow dragging out of and into a CalendarSet area. Commands provide the control necessary to allow or disallow dragging within an area, between two or more areas or from non-plugin objects.

You can allow dropping onto days, events and/or banners. For drags that originate from other sources, the dropping onto days could be conceived as the creation of a new event. Dropping onto events or banners could be thought of as "linking" of some outside information to the event (such as linking a contact to an event).

To allow dragging out of CalendarSet, you must pass an "access code" for the area that is to be dragged from. You must specify at least one code to enable dragging, using CS_SetDrgSrc. 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 below.

In order to allow dropping into CalendarSet, you must pass an "access code" for each type of item (days, events and/or banners) that can be the destination of a drop. You must specify at least one code to enable dropping, using CS_SetDrgDst. As with ***CS_SetDrgSrc***, up to ten codes can be passed for flexibility reasons.

CalendarSet DataType

CalendarSet supports one source data type and three destination data types:

- dragging events and banners
- dropping to days
- dropping to events
- dropping to banners

This **DataType** parameter is used in the commands CS_SetDrgSrc, CS_SetDrgDst, and CS_GetDrgDstTyp. These are the possible values:

Source or destination	Value	Constant
Day (destination)	1	CS_DragDestination_Day
Event (destination)	2	CS_DragDestination_Event
Banner (destination)	3	CS_DragDestination_Banner
Event or banner (source)	2	CS_DragSource_EventBanner

Use CS_GetDrgSrcEvt to determine whether an event or banner was dragged, and which event or banner it was after a drop has completed

What are access “codes”?

The access codes that are passed in CS_SetDrgSrc and CS_SetDrgDst are used to enable dragging between specific drag partners. These drag partners can be the same CalendarSet area, different CalendarSet areas, AreaList Pro plugin areas, text selections from 4D or other applications and external documents.

- Setting at least one source access code to an area will make it draggable.
- Setting at least one destination access code to an area will make it droppable (on the specified object type).

Note: for compatibility reasons the CalendarSet area is draggable and droppable even if it is not set as draggable or droppable in the 4D layout's object properties.

Drag and drop between plugin areas

When dragging from and to CalendarSet or AreaList Pro areas (or within the same area) the source of the drag and the target of the drop are designated as drag and drop partners**.** You need to tell CalendarSet (and AreaList Pro if used) what these partnerships are, and to do this you create **access codes**.

An access code is simply a text code that you create. Let’s say you have a layout that contains two CalendarSet areas: **Cal1** and **Cal2**, and you want to enable items to be dragged from Cal1 and dropped onto Cal2. You might decide on the access code “select”. To enable event and banner dragging you will need two lines of code:

```
CS_SetDrgSrc(Cal1;CS_DragSource_EventBanner;"select")
CS_SetDrgDst(Cal2;CS_DragDestination_Day;"select")
```

You can list up to 10 access codes for each source (events and banners) and destination (days, events or banners). For example, suppose you have four CalendarSet areas on a form: AreaA, AreaB, AreaC and AreaD. You want to allow drag and drop from AreaA to AreaB or AreaC, and from AreaD to AreaC but not AreaB:

1. Create two access codes: “dropB” and “dropC”.

2. Set the access codes for the four areas as follows:

```
CS_SetDrgSrc(AreaA;CS_DragSource_EventBanner;"dropB";"dropC")  // drag to B and C
CS_SetDrgSrc(AreaD;CS_DragSource_EventBanner;dropC")  // drag to C
CS_SetDrgDst(AreaB;CS_DragDestination_Day;"dropB")  // accept drop to days from A
CS_SetDrgDst(AreaC;CS_DragDestination_Day;"dropC")  // accept drop to days from A & D
```

Note: as opposed to AreaList Pro properties where all access codes are passed in the same string (list of codes separated by ‘|’), with CalendarSet each access code is a parameter by itself: "dropB";"dropC".

That’s all you need to do to enable basic drag and drop functionality between two areas with default settings.

When a drag and drop takes place, the drag sender’s plugin code communicates its access codes to the drop receiver’s plugin code. The drop receiver will compare the access codes of the sender to its own codes. If any of the codes match, the drop is allowed. This mechanism allows a number of combinations between several drag and drop partners.

Note: access codes are strictly compared, taking into account case and diacritics.

The following is an example of enabling the dragging and dropping of events within the same CalendarSet area by setting a unique identifier that only enables dragging within this area.

```
// enable drag events to days within this area
vSelfStr:="CalendarSetArea"+String(eCal)  // only allows dragging and dropping within this area
CS_SetDrgSrc(eCal;CS_DragSource_EventBanner;vSelfStr)  // event or banner data type for source
CS_SetDrgDst(eCal;CS_DragDestination_Day;vSelfStr)  // day data type for destination
```

Drag and drop with external objects

External objects are defined in this context as non CalendarSet (or AreaList Pro) plugin areas:

- Dragging from and dropping to 4D fields, variables or any droppable/draggable object
- Dragging from and dropping to 4D text selections (like a text variable content)
- Dragging from and dropping to text selections in other applications (open windows from e.g. a text editor)
- Dragging from external files (no dropping to)

Since external objects obviously don't support access codes they will be potential recipients of any drag from a draggable CalendarSet area, or source of any drop to a droppable CalendarSet area.

Any source access code will make a CalendarSet area draggable to external objects.

The "_external" special access code must be used as a destination access code in order to make a CalendarSet area droppable from external objects:

- "_external" can only be used as a destination access code and is the only way to make a CalendarSet area accept a drop from objects other that CalendarSet or AreaList Pro areas
- "_external" as a destination access code means "drop onto this CalendarSet area from any external object"

We can modify the example above to allow dragging from an external object to area B:

```
CS_SetDrgSrc(AreaA;CS_DragSource_EventBanner;"dropB";"dropC")  //drag to B, C and external objects
CS_SetDrgSrc(AreaD;CS_DragSource_EventBanner;dropC")  //drag to C and external objects
CS_SetDrgDst(AreaB;CS_DragDestination_Day;"dropB";"_external")  //accept drop from A & external objects
CS_SetDrgDst(AreaC;CS_DragDestination_Day;"dropC")  //accept drop from A & D
```

Using the callback method

You can fine-tune your control over the user's drag and drop from the callback method set by CS_SetCallback.

During a drag and drop, this method will be called under two circumstances:

- When a drop is attempted from an external object, while the pointer is still hovering over the CalendarSet destination area: the $0 value returned by the callback will allow the subsequent drop action or not (as with setting access codes to control drag and drop between plugin areas): CS_Action_DragOver.
- After a drop has been performed, whatever the source was (plugin area or external object): CS_Action_Drop.

Note: the callback method will **not** be called with CS_Action_DragOver when the source of the drag and drop is a plugin area, where access codes are used instead to specifically allow drag and drop between areas.

When the drop is completed, the form method/CalendarSet area object (or callback method if previously set by CS_SetCallback) is executed. You can then determine what the last user action was using CS_GetAction (form/object method) or $2 (callback method).

The event 4 (drag) is not reported if the drag ended as drop into the same CalendarSet area.

The source area's last action is 4 (drag), the destination area's last action is CS_Action_Drop.

Action 4 (drag) should no longer be used anyway. Use the CS_Action_Drop callback action instead, or the form/object method On Drop event.

Note: the action reported by CS_GetAction will never be CS_Action_DragOver for an external object source, but the callback will receive this event in $2. See On drag over below.

On drag over

The callback method is called with the CS_Action_DragOver action when an external object is dragged over the area. It is used to allow or reject the drop.

Note: never display a window in the callback while processing this CS_Action_DragOver action, including **TRACE** or **ALERT**: 4D would freeze (ACI0087433).

This method (actually a function in this case) receives two parameters, and must return a result:

- $1 is the destination CalendarSet area reference.
- $2 is the action code, in this case CS_Action_DragOver
- $0 is expected by CalendarSet (only with this action), with either 0 (disallow drop) or 1 (allow drop).

Note: if no callback is set the drop from an external object will always be accepted (provided that "_external" is set as a destination access code for the area). Therefore the callback is the only way to disallow a drop from an external object on a droppable area.

The pointer shape will depend upon this result:

- plus symbol if the drop is allowed: 
- “no entry” sign if the drop is non allowed: 

Note: once an allowed destination has been rolled over, the cursor will stay as “+” even when you subsequently move to a place where the drop is not allowed: the drop will nevertheless be allowed and the On Drop form event will be triggered. This is due to a limitation of 4D, which does not call the plugin again to ask if the drop is allowed unless you leave the area and re-enter it again. In this case ***CS_GetDrgXXX*** will return zeros.

After the drop

When an event or banner is dropped onto a CalendarSet area, the following information is available to you:

- Notification that a drop occurred
- Which item was dragged and dropped (index in array) and what type (event or banner)
- Where the event was dragged from (this area or another area, or another kind of object)
- The type of data that was the recipient of the drop (day, event or banner)
- If the destination type is a day, the date of that day
- If the destination type is an event or banner, the event or banner index within its respective arrays

The processing of the drop event can either be handled in the callback with the CS_Action_Drop action or the form method / CalendarSet area object method with the On Drop event, which is always executed in the context of the destination area / process.The call sequence is:

1. Callback method with CS_Action_Drop in $2

2. Area object method with On Drop form event

3. Form method with On Drop form event

Note: when displaying CalendarSet in an external window there is no form / object method to execute, therefore the callback is the only way to control drag and drop in this context.

The callback method receives two parameters:

- $1 is the destination CalendarSet area reference.
- $2 is the action code, in this case CS_Action_Drop

You can determine which event or banner was dropped using CS_GetDrgSrcEvt.

To determine which object was the source of the drag, call CS_GetDrgSrcArea. This command returns the area reference (a long integer) and the process ID of the source area, whether it is the same CalendarSet area or another CalendarSet area, or an AreaList Pro area (the area reference will be negative in this case).

See also Receiving a drop from a non-CalendarSet Object below.

When dragging to/from another area or a 4D object, that object can either reside in the same window or on another window, which may require use of 4D's process communication commands to take action on the drop.

Note: when dragging and dropping to or from other objects, CalendarSet is only providing a user interface to the drag, and notifying you, the developer, that the drop has occurred. You are responsible for manipulating any arrays or other data structures.

You must call CS_GetDrgDstTyp to determine if the destination of the drop was either a day or event.

- If the destination was a day, CS_GetDrgDstDay is used to determine the destination date.
- If the destination was an event or banner, CS_GetDrgDstEvt is called to determine which event or banner was the destination.

In the case when the sender and the receiver is the same CalendarSet area:

- If the source of the drag is an event and the destination is a day, then CalendarSet will automatically update the event date array.
- For banners, CalendarSet will automatically update the banner start and end date arrays. 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 (e.g. 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).

Note: keep in mind that CalendarSet will update the arrays and refresh the area if (and only if) the drag and drop is within the same area (events or banners to days). Also, dragging onto "out of range" days in the same area will only be allowed if the **UnusedInfo** parameter to CS_Options is set to CS_Unused_GrayedNumBan.

Receiving a drop from a non-CalendarSet Object

As well as dragging between two CalendarSet areas, you can also drag between non-CalendarSet objects and CalendarSet areas — for example, a row or cell selection from AreaList Pro, another 4D object, a text selection in any (drag and drop savvy) application window or a drop from an external document.

Receiving a drop from an AreaList Pro area is identified by CS_GetDrgSrcArea returning a negative value (opposite of the AreaList Pro area reference) in the **SourceArea** parameter and its process in **SourceProcessID**.

If the source of the drag is an external object, the callback (if set for the area) will be triggered twice:

- when dragging over the area ($2=CS_Action_DragOver)
- after the drop ($2=CS_Action_Drop)

In both cases (and in the 4D object/form method after the drop) the **SourceArea** from ***CS_GetDrgSrcArea*** will be zero and the **SourceProcessID** will be:

- the 4D originating object's process if the source is a 4D object
- -1 if the source is a document on disk or another application

Also in both callback calls the content of the dragged and dropped object is a text included in the pasteboard.

Use **GET PASTEBOARD DATA** to analyze the dragged data and allow the drop (CS_Action_DragOver) or process it (CS_Action_Drop).

Allowing the drop from external objects in the callback

When dragging from non-plugin objects, you can use a callback method to “catch” the user action and allow it or not (callback methods are set with the CS_SetCallback command). If no callback is set, the drop will be allowed.

In order for external object drag and drop to be disallowed once an area has been made droppable with the "_external" destination access code, the callback method must be set and handle the CS_Action_DragOver case.

See the External documents example below.

AreaList Pro

The source AreaList Pro area is referenced as a negative value in the **SourceArea** parameter of CS_GetDrgSrcArea (opposite of the AreaList Pro area reference).

Plugin area references use sequential numbering: 1 can be AreaList Pro and CalendarSet area references at the same time, therefore AreaList Pro area #1 will be referenced from CalendarSet's point of view as -1, to differentiate from CalendarSet area #1.

```
C_LONGINT($srcALPArea)
If (Form event=On Drop)  //here we use the object method, no callback set
   CS_GetDrgSrcArea(eCal;$sourceArea;$sourceProcess)
   $srcALPArea:=-$sourceArea  //AreaList Pro's area reference (note the minus sign)
   If ($srcALPArea=ALPappointments)  //is this AreaList Pro area our appointment list?
    //Do something with the appointements
   End if
End if
```

Refer to the AreaList Pro manual regarding accepting a drop in an AreaList Pro area.

4D

You can use the following code in the On Drop event when accepting a drop from a 4D object:

```
DRAG AND DROP PROPERTIES($srcObject;$srcElement;$srcProcess)
```

Note: $srcObject is **Nil** if the source 4D object has Automatic Drag enabled.
$srcObject is also **Nil** if it comes from a different application (or 4D instance).

Then you can use the following code to check what has been dropped:

```
ARRAY TEXT($4Dsignatures;0)
ARRAY TEXT($nativeTypes;0)
ARRAY TEXT($formatNames;0)
GET PASTEBOARD DATA TYPE($4Dsignatures;$nativeTypes;$formatNames)
```

Note: the On Drop event code will work correctly after the drop when used in an area’s object method but in an event callback the form event is zero and drag & drop properties from 4D will not function. However the pasteboard can still be analyzed in the callback with both actions CS_Action_DragOver and CS_Action_Drop.

External documents

After Allowing the drop from external objects in the callback, the CS_Action_Drop action or the On Drop event are used to open the document according to the pathname retrieved from the pasteboard, then process it.

In the following example (available in the CalendarSet demonstration database) we import events from a XML document. We set the callback and allow drop from external documents in the CalendarSet area object method:

```
Case of
   : (Form event=On Load)
      CS_SetDrgDst (xCalendar;CS_DragDestination_Day;"_external")  // destination days, external only
      CS_SetCallback (xCalendar;"CSEventCallbackDD")  // callback method
End case
```

Then we allow the drop in the callback if the external source is a XML type document, and if such is the case we process it to create our events.

```
C_LONGINT($1;$2;$0;$i;$elements)  // $0 for CS_Action_DragOver only
C_DATE($date)
C_TEXT($path;$ref;$xml_Tmp;$xml_Tmp2;$description;$participant;$hour)
C_BLOB($data)
Case of
   : ($2=CS_Action_DragOver)
      $0:=0  // disallow drop unless it is an external XML file
      GET PASTEBOARD DATA("com.4d.private.file.url";$data)  // gets file pathname if external document
      If (OK=1)  // something valid in the pasteboard
         $path:=Get file from pasteboard(1)  // first file
         If ($path="@xml")  // is it a XML document?
            $0:=1  // allow drop
         End if
      End if
   : ($2=CS_Action_Drop)  // was tested above so we know it is a valid XML document
      GET PASTEBOARD DATA("com.4d.private.file.url";$data)  // gets file pathname again (could be a global)
      $path:=Get file from pasteboard(1)  // first file
      $ref:=DOM Parse XML source($path;False)
      If (OK=1)
         CS_GetDrgDstDay ($1;$date)  // day where dropped
         $elements:=DOM Count XML elements($ref;"event")
         For ($i;1;$elements)  // get each item
            $xml_Tmp:=DOM Get XML element($ref;"event";$i;$data)
            $xml_Tmp2:=DOM Get XML element($xml_Tmp;"participant";1;$participant)
            $xml_Tmp2:=DOM Get XML element($xml_Tmp;"description";1;$description)
            $xml_Tmp2:=DOM Get XML element($xml_Tmp;"hour";1;$hour)
            APPEND TO ARRAY(vDDRecDates;$date)
            APPEND TO ARRAY(vDDRecDesc;$participant+" : "+$description)
            APPEND TO ARRAY(vDDRecTimes;Time($hour))
         End for
         DOM CLOSE XML($ref)
         Area_Refresh ($1)  // inform CalendarSet
      End if
End case
```

Commands

CS_GetDrgArea

(AreaRef; DestArea; DestProcessID)

Parameter	Type	Description
→ AreaRef	longint	ara reference of CalendarSet object on form
← DestArea	longint	ID of the area the item was dragged to
← DestProcessID	longint	process ID of the **DestArea**

This command is obsolete and should no longer be used. Use CS_SetDrgSrcArea in the destination area instead.

CS_GetDrgDstDay

(AreaRef; DestDate)

Parameter	Type	Description
→ AreaRef	longint	area reference of the CalendarSet area
← DestDate	date	day which received the last drop

If the destination of the last drop was a day (See CS_GetDrgDstTyp), use ***CS_GetDrgDstDay*** to determine the destination date. The **AreaRef** parameter must be the destination (receiver) area of a drop.

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

Example

```
  //eDestCal object method (On Drop event)
C_LONGINT($sourceArea;$sourceProcess)
C_INTEGER($actionCode;$type)
C_DATE(vDate)
CS_GetAction(eDestCal;$actionCode)  //determine user action
Case of
   :($actionCode=CS_Action_Drop)  //user dropped an event or banner
      CS_GetDrgSrcArea(eDestCal;$sourceArea;$sourceProcess)
      If($sourceProcess=Current process)  //was the drag from an area in this process?
         If ($sourceArea=eDestCal )  //if dragged within the same area
            CS_GetDrgDstTyp(eDestCal;$type)  //get the type of data that was destination of the drag
            If($type=CS_DragDestination_Day)  //if dragged onto a day
               CS_GetDrgDstDay(eDestCal;vDate)  //get day's date
            End if
         End if  //$sourceProcess=Current process
      Else  //dropped from a different process
         VARIABLE TO VARIABLE($sourceProcess;vDate;vDate)  //send the date to the source process
      End if
End case
```

CS_GetDrgDstEvt

(AreaRef; DestType; EventOrBannerIndex)

Parameter	Type	Description
→ AreaRef	longint	area reference of the CalendarSet area
→ DestType	integer	day, event or banner
← EventOrBannerIndex	integer	index into corresponding array element

If the destination of the last drop was an event or banner (see CS_GetDrgDstTyp), use this command to determine which event or banner was the destination. The **AreaRef** parameter is the destination (receiver) area of a drop.

DestType — Integer, 1 or 2. Values are shown below:

Type	Value	Constant
Event	1	CS_Type_Event
Banner	2	CS_Type_Banner

Note: the values above are different from the CS_SetDrgDst and ***CS_GetDrgDstTyp*** data types.
The constants used are the same as in CS_GetEvtSelect and CS_SetEvtSelect

EventOrBannerIndex — Integer. Index within the array that was passed to CalendarSet via either CS_SetArray or CS_SetBanrArray.

Example

In this example we use the callback method set for the area, to handle receiving a drop from the eSrcCal area in the same process (where banner arrays are prefixed by “src”).

We assume that our area has been set droppable to any data type, using ***CS_SetDrgDst***.

If a banner is dragged from the source CalendarSet area and dropped on a banner in our area, we decided that it should be removed from the source area (for the example).

```
  // eDestCal callback method
C_LONGINT($1)  // area reference.
C_LONGINT($2)  // action code
C_LONGINT($sourceArea;$sourceProcess)  // source area reference (where drag was initiated) ; its process
C_INTEGER($type;$sourceIndex;$destIndex)  // data type ; array index
Case of
   :($2=CS_Action_Drop)  // drop received
      CS_GetDrgSrcArea($1;$sourceArea;$sourceProcess)
      If($sourceArea=eSrcCal) & ($sourceProcess=Current process)  // drag from eSrcCal in the same process
      CS_GetDrgSrcEvt($1;$type;$sourceIndex)  // no need for date here, we can use only 3 parameters
         If($type=CS_Type_Banner)  // a banner was dragged
            CS_GetDrgDstEvt($1;$type;$destIndex)  // where was it dropped?
            If($type=CS_Type_Banner)  // drop was on a banner
               DELETE FROM ARRAY(srcStartDateArray;$sourceIndex)  // remove the banner from all arrays
               DELETE FROM ARRAY(srcEndDateArray;$sourceIndex)
               DELETE FROM ARRAY(srcTextArray;$sourceIndex)
```

```
            DELETE FROM ARRAY(srcFontArray;$sourceIndex)
            DELETE FROM ARRAY(srcSizeArray;$sourceIndex)
            DELETE FROM ARRAY(srcStyleArray;$sourceIndex)
            DELETE FROM ARRAY(srcForeColorArray;$sourceIndex)
            DELETE FROM ARRAY(srcBackColorArray;$sourceIndex)
            Area_Refresh ($sourceArea)  //inform CalendarSet
         End if  //drop was on a banner
      End if  //a banner was dragged
   End if  //drag from eSrcCal in the same process
End case
```

CS_GetDrgDstTyp

(AreaRef;DestDataType)

Parameter	Type	Description
➞ AreaRef	longint	area reference of the CalendarSet area
➞ DestDataType	integer	day, event or banner

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

DestDataType — Integer, 0 to 3. Indicates what type of data was the destination of the last drop. Values are shown below:

Destination	Value	Constant
No destination	0	CS_DragDestination_None
Day	1	CS_DragDestination_Day
Event	2	CS_DragDestination_Event
Banner	3	CS_DragDestination_Banner

See the example to CS_GetDrgDstDay.

CS_GetDrgSrcArea

(AreaRef; SourceArea; SourceProcessID)

Parameter	Type	Description
→ AreaRef	longint	ara reference of CalendarSet object on form
← SourceArea	longint	ID of the area the item was dragged from
← SourceProcessID	longint	process ID of the **SourceArea**

Use ***CS_GetDrgSrcArea*** to determine the source area of the last drop. The **AreaRef** parameter is the destination (receiver) area of a drop.

This command is called from the destination area's object/form method (or callback method receiving a action code of CS_Action_Drop in $2).

SourceArea — Long integer. This parameter is the area reference of the area that is the source of the drag.

SourceProcessID — Long integer. This parameter contains the process ID in which the window and source area reside.

Note: if the **SourceProcessID** is different from the current process, you will need to use 4D's process communication commands to exchange values with the process that contains the source area.

See the examples to CS_GetDrgDstDay and CS_GetDrgDstEvt.

CS_GetDrgSrcEvt

(AreaRef; DataType; EventIndex; StartDate)

Parameter	Type	Description
→ AreaRef	longint	area reference of the CalendarSet area
← DataType	integer	type of item dragged
← EventOrBannerIndex	integer	index into corresponding array element
← StartDate	date	starting date of the drag

Use CS_GetDrgSrcEvt to determine whether an event or banner was dragged and which event or banner it was.

The **AreaRef** parameter is the destination (receiver) area of a drop. This command is called from the destination area's object/form method (or callback method receiving a action code of CS_Action_Drop in $2).

DataType — Integer, 1 or 2. Values are shown below:

Type	Value	Constant
Event	1	CS_Type_Event
Banner	2	CS_Type_Banner

EventOrBannerIndex — Integer. This parameter is an index within the array that was passed to CalendarSet via either CS_SetArray or CS_SetBanrArray .

StartDate — Date. For events, this date will the event's date passed in CS_SetArray. For banners, this date will be the starting point of the drag, which will be a value between the banner's start and end dates, inclusively.

See the example to CS_GetDrgDstEvt.

CS_SetDrgDst

(AreaRef; DestDataType; DstCode1; DstCode2; ... ; DstCode10)

Parameter	Type	Description
→ AreaRef	longint	area reference of the CalendarSet area
→ DestDataType	integer	data type to be received
→ DstCodeN	text	access code(s) to allow drop

CS_SetDrgDst is used to enable dropping into a CalendarSet destination area, by setting the access codes for the destination of the drop. Please read the section What are access "codes"? for more information.

This command must be called before a drag is initiated (usually in the On load form event processing).

The **AreaRef** parameter is the destination (receiver) area of a drop.

DestDataType — Integer, 1 to 3. Indicates what type of data can be a destination of a drop:

Destination	Value	Constant
Day	1	CS_DragDestination_Day
Event	2	CS_DragDestination_Event
Banner	3	CS_DragDestination_Banner

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

DstCode — Text. The **DstCode** can be any value (other than an empty string), such as "MonthDrag", "WeekDrag", "ALPDrag", etc.

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 an AreaList Pro area.

CalendarSet performs the following logic when the drag and drop takes place: the destination codes that were given in **DstCode1**, **DstCode2**, etc. are compared to the sender's source codes, previously set with CS_SetDrgSrc.

If any of the codes match, the drop is enabled.

In addition, the "_external" special access code must be used as a destination access code in order to make a CalendarSet area droppable from external objects. See Allowing the drop from external objects in the callback.

- "_external" can only be used as a destination access code and is the only way to make a CalendarSet area accept a drop from objects other that CalendarSet or AreaList Pro areas
- "_external" as a destination access code means "drop onto this CalendarSet area from any external object"

Examples

```
 //enable dragging events to events within this area
vSelfStr:="CalendarSetArea"+String(eCal)  //only allows dragging and dropping within this area
CS_SetDrgSrc(eCal;CS_DragSource_EventBanner;vSelfStr)  //event data type for source
CS_SetDrgDst(eCal;CS_DragDestination_Event;vSelfStr)  //event data type for destination

 //enable dragging from another CalendarSet area (to days or events) or an external object (to days or banners)
CS_SetDrgDst(eMonthCal;CS_DragDestination_Day;"WeekToMonth";"_external")  //allow dragging into days
CS_SetDrgDst(eMonthCal;CS_DragDestination_Event;"WeekToMonth")  //allow dragging into events
CS_SetDrgDst(eMonthCal;CS_DragDestination_Banner;"_external")  //allow dragging into banners
```

CS_SetDrgSrc

(AreaRef; SourceDataType; SrcCode1; SrcCode2; ... ; SrcCode10)

Parameter	Type	Description
→ AreaRef	longint	area reference of the CalendarSet area
→ SourceDataType	integer	type of item dragged
→ SrcCodeN	text	access code(s) to allow drag

CS_SetDrgSrc is used to enable dragging out of a CalendarSet source area, by setting the access codes for the source of the drag. Please read the section What are access “codes”? for more information.

This command must be called before a drag is initiated (usually in the On load form event processing).

The **AreaRef** parameter is the source (sender) area of a drag.

SourceDataType — Integer, 2. Currently, only events and banners can be dragged from CalendarSet:

Source	Value	Constant
Event or banner	2	CS_DragSource_EventBanner

SrcCode — Text. The **SrcCode** can be any value (other than an empty string), such as “MonthDrag”, “WeekDrag”, “ALPDrag”, etc.

The code should be the same as what is 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 an AreaList Pro area.

CalendarSet performs the following logic when the drag and drop takes place: the source codes that were given in **SrcCode1**, **SrcCode2**, etc. are compared to the receiver's destination codes, previously set with CS_SetDrgDst.

If any of the codes match, the drop is enabled.

In addition, any source access code will make a CalendarSet area draggable to external objects.

See the examples to CS_SetDrgDst.

Note: events and banners, though set by the same **SourceDataType** value, behave a bit differently.

To be draggable, both events and banners must be selectable (and a source access code must be present), but the default values differ:

- Default for event: not selectable ($2 of CS_SetEventOpts)
- Default for banner: selectable ($2 of CS_SetBanrOpts)

Popup Commands

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

Using the Color Selection Popup

A color selection popup is available to aid the development of an interface for selecting color of calendar data, using the CalendarSet Color table.

The color popup includes the CM_SetColor command for setting the current color, and the CM_GetColor command to get the user selected color.

Using the Icon Selection Popup

An icon selection popup is included to display pictures from the 4D library in a palette. This provides an elegant interface tool for allowing user selection of icon data for a calendar.

The icon popup includes the IM_SetArray command for specifying what icons to include in the popup and the size of the icons, the IM_SetSelect command for setting the current popup item, and the IM_GetSelect command for getting the user selected item.

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.

The date popup command to set the current value is MM_SetDate, and the command to get the value selected by the user is MM_GetDate. Two designs are available: one is MacOS oriented (legacy style), the other more of a Windows type (see Alternate “Windows” date popup), but you can use any option on both platforms.

The time popup command to set the current value is TM_SetTime, and the command to get the value selected by the user is TM_GetTime.

Note: both date and time popups will close on Escape or Cmd / Ctrl-dot (cancel), Enter, Shift-Enter, Return, Shift-Return, Tab or Shift-Tab (accept), Del (clear) keys.

See also MM_SetOptions and TM_SetOptions regarding click options to validate the popups.

Commands

%CM_PopArea

%***CM_PopArea*** is a plugin area which allows you to implement a color palette popup on a 4D form. This popup allows the user to select a color from the Color table.

Area_SetEnable

(AreaRef; Enabled)

Parameter	Type	Description
➞ AreaRef	longint	area reference of the popup area object on form
➞ Enabled	integer	disable (0) or enable (1) the popup object

Area_SetEnable allows you to procedurally enable / disable any of the four (colors, icons, dates, times) popup type objects available in the CalendarSet plugin.

Note: this command does not affect a regular CalendarSet area.

Enabled — Integer. Use this parameter to specify whether you wish to disable (0) or enable (1) the popup.

Once disabled, the popup is not displayed when its icon is clicked. There is no need to call Area_Refresh after ***Area_SetEnable***.

Examples

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

CM_GetColor

(AreaRef; ColorToGet)

Parameter	Type	Description
➞ AreaRef	longint	area reference of the %***CM_PopArea*** object on form
⬅ ColorToGet	integer	currently selected color in the popup object

CM_GetColor allows you to retrieve the color that the user selected the last time the color popup was used.

ColorToGet — Integer. Use this parameter to identify what color the user selected. Refer to the Color table for possible values.

Example

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

CM_SetColor

(AreaRef; ColorToSet)

Parameter	Type	Description
➞ AreaRef	longint	area reference of the %***CM_PopArea*** object on form
➞ ColorToSet	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. Refer to the Color table for possible values.

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; CS_Color_Red)
```

%IM_PopArea

%***IM_PopArea*** allows you to implement an icon popup menu.

The icons that are displayed by the popup must be stored in the picture library.

IM_GetSelect

(AreaRef; CurrentSelectedItem)

Parameter	Type	Description
➞ AreaRef	longint	area reference of the %***IM_PopArea*** object on form
⬅ CurrentSelectedItem	integer	index number 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 4D picture library 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 4D picture library ID for the currently selected icon in the eIcon object
IM_GetSelect(eIcon;vIndex)
[Calendar]Icon ID:=aIcons{vIndex}
```

IM_SetArray

(AreaRef; ArrayName; IconSize; NumColumns)

Parameter	Type	Description
→ AreaRef	longint	area reference of the %***IM_PopArea*** object on form
→ IconArray	text	name of the array of picture library numbers for each icon
→ IconSize	Integer	size to use for the icon
→ NumColumns	integer	number of columns of icons to display in the popup

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

IconArray — Text (name of an integer array). This array contains the 4D picture library IDs of the icons to display. The icons specified in **IconArray** must be stored in the picture library.

IconSize — Integer. This parameter is the size of the icon to use. The following values are allowed:

Mode	Value	Constant
Small Icons (16 x 16)	16	CS_Icon_Small
Large Icons (32 x 32)	32	CS_Icon_Large

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 aIcons
// the icon object is eIcon, and the icons should be displayed as large icons
// the icon popup should have an equal number of rows and columns
IM_SetArray(eIcon;"aIcons";CS_Icon_Large;0)
```

IM_SetSelect

(AreaRef; CurrentSelectedItem)

Parameter	Type	Description
→ AreaRef	longint	area reference of the %***IM_PopArea*** object on form
→ CurrentSelectedItem	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 eIcon to display the icon saved in [Calendar]Icon ID
$ID_Index:=Find in array(aIcons;[Calendar]Icon ID)
If($ID_Index>0)  // make sure the icon id is in the array
   IM_SetSelect(eIcon;$ID_Index)
End if
```

%MM_PopArea

%***MM_PopArea*** allows you to implement a calendar popup.

MM_GetDate

(AreaRef; DateToGet)

Parameter	Type	Description
→ AreaRef	longint	area reference of the %***MM_PopArea*** object on form
← DateToGet	date	currently selected date in the calendar popup

MM_GetDate allows you to retrieve the date that the user selected the last time the MM popup was used.

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

Example

```
  //date popup object method
If(Form event=On Clicked)
   MM_GetDate(ePopDate;vDate)
   [Invoice]Invoice Date:=vDate
End if
```

MM_SetDate

(AreaRef; DateToSet)

Parameter	Type	Description
→ AreaRef	longint	area reference of the %***MM_PopArea*** object on form
→ 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).

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

Example

```
  //date popup object method ("only if modified" checkbox is turned off)
If(Form event=On Load)
   If(Record number([Invoice])=New record)  //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_SetOptions

(AreaRef; popIndicator; useDoubleClick; design; colors)

Parameter	Type	Description
→ AreaRef	longint	area reference of the %***MM_PopArea*** object on form
→ popIndicator	integer	popup indicator: arrows (0), none (1), calendar picture (2)
→ useDoubleClick	integer	close on single click (0) or double click (1)
→ design	integer	MacOS style (0) or Windows style (1)
→ colors	text	calendar colors

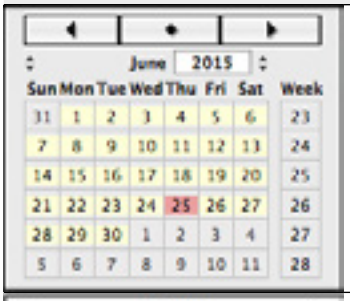
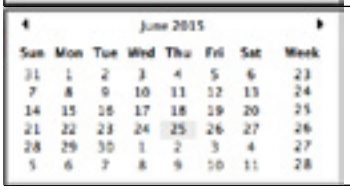
MM_SetOptions is used to configure the calendar popup object specified by AreaRef.

popIndicator — Integer, 0 to 2. This parameter lets you set the 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 plugin object. Possible values are:

Symbol	Value	Constant
Popup arrow(s)	0	CS_MM_PopArrows
None (transparent)	1	CS_MM_PopNone
Calendar picture (default)	2	CS_MM_PopPicture

useDoubleClick — Integer, 0 or 1. If this parameter is set to 0 (default) the popup will close on single click. Set it to 1 if you want it to close on double click.

design — Integer, 0 or 1. This parameter defines the look of the popup calendar. Both designs are available on both platforms:

Design	Value	Constant
	0	CS_MM_DesignMac
	1	CS_MM_DesignWindows

colors — Text. 8 ARGB colors separated with "|" to be used by the popup.

- First 5 colors define object backgrounds: active month, inactive month, selected date, current date, current selected date
- Next 2 colors define foreground: numbers in active month, numbers in inactive month
- 8th value is the popup background color; it needs a non-zero alpha channel to be set, e.g. #FFE9F1FF instead of #E9F1FF

Default values are:

- “#00FFFFDD|#00EEEEEE|#00EEAAAA| #00FF8888|#00FF5555|#00000000| #00444444|#00CCCCCC” for the MacOS (default) calendar look, and:
- “#FFFFFFFE|#FFFFFFFE|#00EEEEEE| #00FF8888|#008F8F8F|#00000000| #00444444|#FFFFFFFC” for the Windows type popup (according to **design**).

Examples

```
//hide the indicator
MM_SetOptions(eDatePop;CS_MM_PopNone)
//display calendar picture as indicator
MM_SetOptions(eDatePop;CS_MM_PopPicture)
```

%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 and TM_GetTime are not time variables, but rather text variables that are the names of the time variables.

TM_GetTime

(AreaRef;TimeToGet)

Parameter	Type	Description
➞ AreaRef	longint	area reference of the %***TM_PopArea*** object on form
⬅ TimeToGet	text	time selected last time popup clicked (variable name)

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

TimeToGet — Text. 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
```

TM_SetOptions

(AreaRef; popIndicator; useDoubleClick)

Parameter	Type	Description
➞ AreaRef	longint	area reference of the %***TM_PopArea*** object on form
➞ popIndicator	integer	popup indicator: arrows (0), none (1), time picture (2)
➞ useDoubleClick	integer	close on single click (0) or double click (1)

TM_SetOptions is used to configure the calendar popup object specified by **AreaRef**.

popIndicator — Integer, 0 to 2. This parameter lets you set the 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 plugin object. Possible values are:

Symbol	Value	Constant
Popup arrow(s)	0	CS_TM_PopArrows
None (transparent)	1	CS_TM_PopNone
Time picture (default)	2	CS_TM_PopPicture

useDoubleClick — Integer, 0 or 1. If this parameter is set to 0 (default) the popup will close on single click on minutes. Set it to 1 if you want it to close on double click on minutes.

Note: in both cases the popup will close on double click on hours.

Examples

```
//hide the indicator
TM_SetOptions(eTimePop;CS_TM_PopNone)
//display calendar picture as indicator
TM_SetOptions(eTimePop;CS_TM_PopPicture)
```

TM_SetTime

(AreaRef;TimeToSet)

Parameter	Type	Description
➞ AreaRef	longint	area reference of the %***TM_PopArea*** object on form
➞ TimeToSet	text	time to preselect when popup clicked (variable name)

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 — Text. 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")
```

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 current workstation OS settings.

To obtain the correct day name for the currently selected location, use CS_GetDayName. Use CS_GetMonthName to get the correct month name.

Array Intersection

You can intersect two arrays to create a third array using FS_ArrayIntrsct.

Commands

Area_Refresh

(AreaRef)

Parameter	Type	Description
➞ AreaRef	longint	area reference of the CalendarSet area

Area_Refresh will redraw any of the plugin area objects in the CalendarSet plugin, using all of the settings that were changed.

This command is generally not necessary. For example you don't need it for the popup areas. When you change any setting (including options / current value of popup areas), CalendarSet will handle it automatically.

It can, however, be useful when modifying an array displayed by a calendar or other object type, e.g. a picture popup array. But you can also simply reset the respective array (CS_SetArray, CS_SetBanrArray, IM_SetArray).

Example

```
// after adding an item to the arrays being displayed in the calendar
// object eCalendar, update the object
AddCalenItem(vDate;vText)
Area_Refresh(eCalendar)
```

CS_GetDayName

(DayNumber; Abbreviated; DayName)

Parameter	Type	Description
➞ DayNumber	integer	day number to get the name of
➞ Abbreviated	integer	get abbreviated day name
← DayName	text	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 OS that is being used. For example,Tuesday abbreviated in the United States is “Tue”.

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

Example

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

CS_GetMonthName

(MonthNumber; Abbreviated; MonthName)

Parameter	Type	Description
→ MonthNumber	integer	month number to get the name of
→ Abbreviated	integer	get abbreviated month name
← MonthName	text	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 OS that is being used. For example, “February” is abbreviated in the United States as “Feb”.

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

Examples

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

CS_GetPicture

(AreaRef) ➞ picture

Parameter	Type	Description
➞ AreaRef	longint	area reference of the CalendarSet area
⬅ picture	picture	vector picture of the area

CS_GetPicture returns a PDF (MacOS) or EMF (Windows) vector picture of the specified area. This feature can be useful for web publishing or offscreen printing, amongst others. It can be used for offscreen areas as well.

Example

```
//get a picture of the current calendar into a variable
C_PICTURE(vPict)
vPict:=CS_GetPicture (eCalArea)
```

CS_SimClick

(AreaRef; whereX; whereY)

Parameter	Type	Description
➞ AreaRef	longint	area reference of the CalendarSet area
➞ whereX	integer	horizontal relative coordinate of the click
➞ whereY	integer	vertical relative coordinate of the click

CS_SimClick posts a click event in an area, either on screen (same as 4D's **POST CLICK**) or offscreen.

For onscreen areas the **whereX** and **whereY** coordinates are relative to the window.

Example

```
//post a click in a offscreen calendar area at coordinates 100 (horizontal), 50 (vertical)
CS_SimClick (eCalArea;100;50)
```

FS_ArrayIntrsct

(CompareArray; SourceArray; DestArray)

Parameter	Type	Description
➞ CompareArray	integer	array of indices into source
➞ SourceArray	array	array to extract items from
⬅ DestArray	array	array to place intersected items into

FS_ArrayIntrsct allows you to build an array based upon the intersection of an Integer array of indices and any other array.

The elements of **SourceArray** with indices from **CompareArray** are copied into **DestArray**.

This command is useful with the CalendarSet routine CS_GetSelItems. 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.

Note: “any type” actually means compatible types: the array must be one-dimensional and not an array of pointers or BLOBs.

DestArray — Array (same type as **SourceArray**). The results of the intersection are placed into this array.

Example

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

If(Form event=On Clicked)
   CS_GetAction(eCal;vAction)
   If(vAction=CS_Action_ClickOnDay)  // did the user select a day or days?
      // get the events on this day or days
      CS_GetSelItems(eCal;CS_Sel_Events;aIndex)
      // build an array with the event text for events on the selected days
      FS_ArrayIntrsct(aIndex;aEventText;aSelDayEvts)
   End if  // CS_Action_ClickOnDay
End if  // On Clicked
```

Util_SetDate

(DateToSet; Month; Day; Year)

Parameter	Type	Description
← DateToSet	date	date to be set
→ Month	integer	month number of the date
→ Day	integer	day number of the date
→ Year	integer	year of the date

This command is deprecated. Use 4D's **Add to date** function:

```
$date:=Add to date (!00/00/0000!;$years;$months;$days)
```

Note: be careful, the years parameter is before months in this syntax.

Offscreen Commands

CalendarSet allows offscreen areas, which are not displayed (neither in a form nor in an external window).

Offscreen areas are created with the CS_NewOSArea command, which returns the area reference to be used as the first parameter to most CalendarSet commands.

An offscreen area can be populated and formatted just like a visible area, then it can be converted to a vector picture using CS_GetPicture. Clicks on the offscreen area can be emulated with CS_SimClick.

See the CalendarSet Demonstration database for an example of creating an offscreen area programmatically.

Don't forget to destroy your offscreen area with CS_DeleteOSArea once it is no longer needed.

Commands

CS_DeleteOSArea

(AreaRef)

Parameter	Type	Description
➞ AreaRef	longint	area reference of the offscreen CalendarSet area to delete

CS_DeleteOSArea deletes an offscreen area (previously created with ***CS_NewOSArea***) from memory.

Make sure that you always delete offscreen areas once you're done with them.

Example

```
// delete an offscreen area
CS_DeleteOSArea($area)  // $area is not an on-screen plugin area, but is from CS_NewOSArea
```

CS_NewOSArea

(areaName; width; height) ª AreaRef

Parameter	Type	Description
➞ areaName	text	name of the offscreen area to create
➞ width	integer	offscreen area width
➞ height	integer	offscreen area height
⬅ AreaRef	longint	area reference of the new offscreen CalendarSet area

CS_NewOSArea creates an offscreen area with the specified name, width and height (at position 0, 0) and returns its area reference.

Example

```
// create an offscreen 800 x 600 area and set its range
C_LONGINT($area)
$area:=CS_NewOSArea("MyOffscreenArea";800;600)
CS_SetRange($area;$dateFrom;$dateTo)  // $area is not an on-screen plugin area, but is from CS_NewOSArea
```

CalendarSet in a plugin Window

Using CalendarSet in a plugin Window

You can display CalendarSet in an independent, resizable window.

This type of window is created using the 4D **Open external window** command.

The syntax of the command is as follows:

Open external window(left;top;right;bottom;type;title;plugInArea) ➞ AreaRef

Parameter	Type	Description
➞ left	longint	global left coordinate of window contents area
➞ top	longint	global top coordinate of window contents area
➞ right	longint	global right coordinate of window contents area
➞ bottom	longint	global bottom coordinate of window contents area
➞ type	longint	window type
➞ title	text	title of window
➞ plugInArea	text	plugin object name: use "_CS_Area" for CalendarSet
⬅ AreaRef	longint	area reference of plugin area, used by CalendarSet commands

After opening the plugin 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 object method or form method, a callback method is used.

This is a method you specify with CS_SetCallback, 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)
vDay1ThisMth:= Add to date(!00/00/0000!;Year of(Current date);Month of(Current date);1)
vDay1NextMth:= Add to date(vDay1ThisMth;0;1;0)
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;aItems;[Appts]Item Font;\
   aFonts;[Appts]Size;aSizes;[Appts]Styles;aStyles;[Appts]Color; aColors)
 //and display the data in the CalendarSet plugin area named "eMonth"
 //open a CalendarSet plugin window
eMonth:=Open external window(10;50;400;450;4;"plugin Window Example";"_CS_Area")
 //setup the CalendarSet area
CS_SetArray (eMonth;"aDates";"aItems";"aFonts";"aSizes";"aStyles";"aColors")
 //highlight number only, multiple cells (contiguous), grayed cells with info,
 //month prefix, Sunday start, extend frame
CS_Options (eMonth;CS_Highlight_Number;CS_DaySelect_Multiple;CS_Unused_GrayedNumBan;\
   CS_MonthOnFirstOn;CS_FirstDaySun;CS_Frame_Extended)
 //allow event selection, word-wrap event text, mark events with bullet
CS_SetEventOpts(eMonth;CS_EventSelect_Single;CS_WordWrap_On;CS_Marker_Bullet)
 //allow banner selection, allow banner resizing
CS_SetBanrOpts(eMonth;CS_Banner_Select;CS_Banner_Resize)
 //grid color blue, background color very light gray
CS_SetCalColor(eMonth;CS_Color_Blue;241)
CS_SetCallback(eMonth;"HandleCalClick")
 //Project method HandleCalClick
C_LONGINT($1;$CalArea)  //CalendarSet object reference
C_INTEGER($2;$Action)  //type of user action
$CalArea:=$1  //reassign received parameters for better readability
$Action:=$2
Case of
   :($Action=CS_Action_ClickOnDay)  //user click on a day
      CS_GetSelect ($CalArea;vDateSt;vDateEn;vContig;aDays)
      ModDayInfo ($CalArea;vDateSt)  //add/modify the info for that day
   :($Action=CS_Action_WindowClosed)  //user click in plugin window close box or the hosting form is closed
      SaveCalendar  //save any changes
   :($Action=CS_Action_ClickOnEventBanner)  //clicked on an event or bannerv
   :($Action=CS_Action_BannerResize)  //resized a banner
   :($Action=CS_Action_Drop)  //dropped an event or banner (or a non-CalendarSet object)
    //handle the drag and drop here
   :($Action=CS_Action_DragOver)  //dragging a non CalendarSet/AreaList Pro object over the area
      $0:=1  //allow drop
End case
```

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.

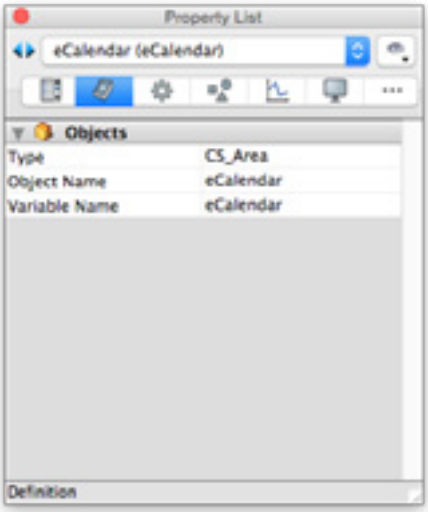
These examples use CalendarSet commands to configure the behavior of an object, rather than the Advanced Properties Dialog. You can reduce the programming effort by using the Advanced Properties Dialog. Please read the section Using the Advanced Properties Dialog for more information.

Note: all the examples below, and many others, are available in the CalendarSet Demonstration database.

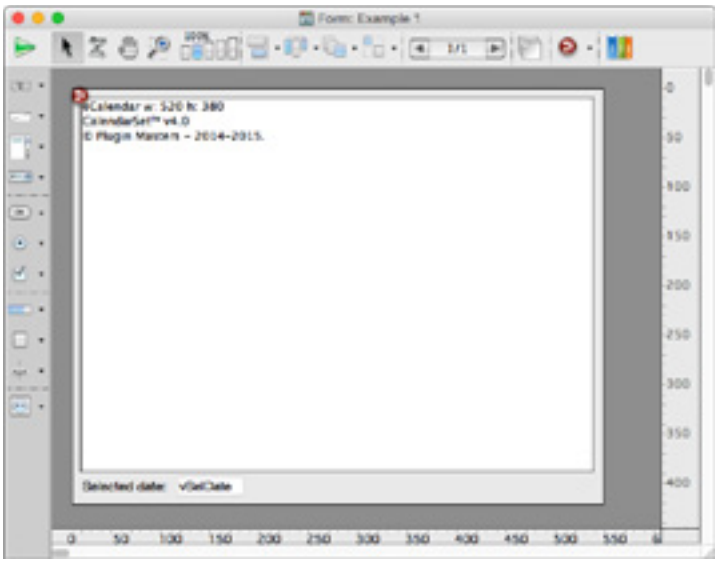
Example 1 — Create a Simple Calendar

For this example, we will create a simple form 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 form and draw the CalendarSet plugin object. We'll name the object eCalendar.



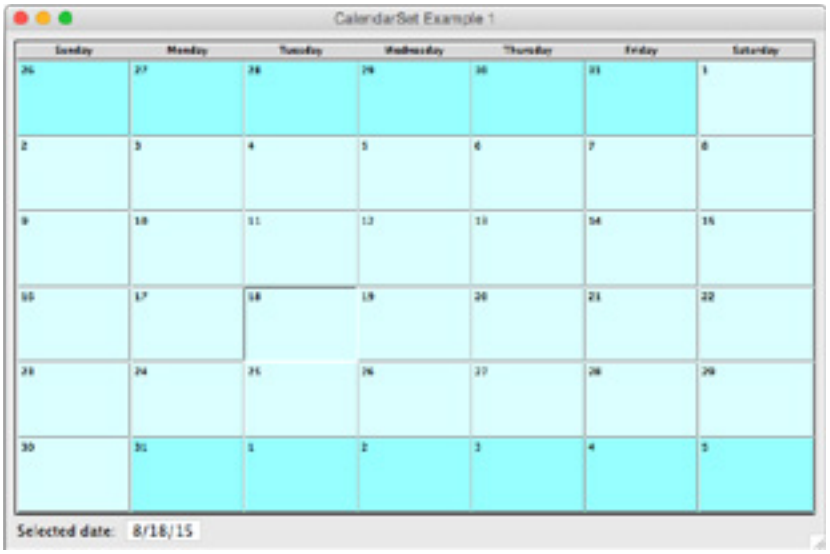
Our form now looks something like this:



The object method for the CalendarSet object is:

```
If(Form event=On Clicked)
   CS_GetAction (Self->;vAction)
   If(vAction=CS_Action_ClickOnDay)  // clicked on a day
      CS_GetSelect(Self->;vSelDate;vSelDateE;isContig;aDates)
   End if
End if
```

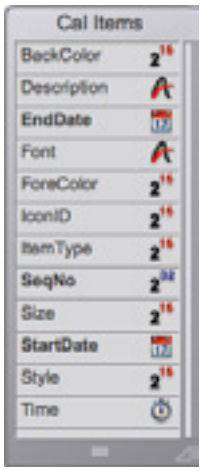
The form will appear like this in the User or Runtime environment:



Notice that we've used the default configuration options, and the only programming required is a couple of commands to detect a click on a day, then determine what day the user has selected.

Example 2 — Display Database Data in a Calendar

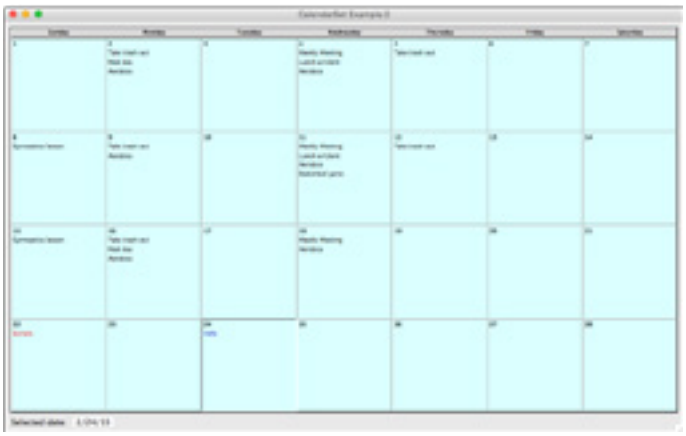
Modify the previous example to display information from the [Cal Items] file on the calendar. The file structure for this example is:



The On Load section of the modified object method for the CalendarSet object is:

```
Case of
   : (Form event=On Load)
      vSelDate:=!00/00/00!
      CS_SetRange(Self->;!02/01/2015!;!02/28/2015!)
      QUERY([Cal Items];[Cal Items]ItemType=1)  // Items of type 1 are text items for this demo
      SELECTION TO ARRAY([Cal Items]StartDate;vTextDate;[Cal Items]Description;vTextDesc;\
         [Cal Items]Font;vTextFont;[Cal Items]Size;vTextSize;[Cal Items]Style;vTextStyle;\
         [Cal Items]ForeColor;vTextColor)
      CS_SetArray(Self->;"vTextDate";"vTextDesc";"vTextFont";"vTextSize";"vTextStyle";"vTextColor")
(...)
End case
```

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



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 object method.

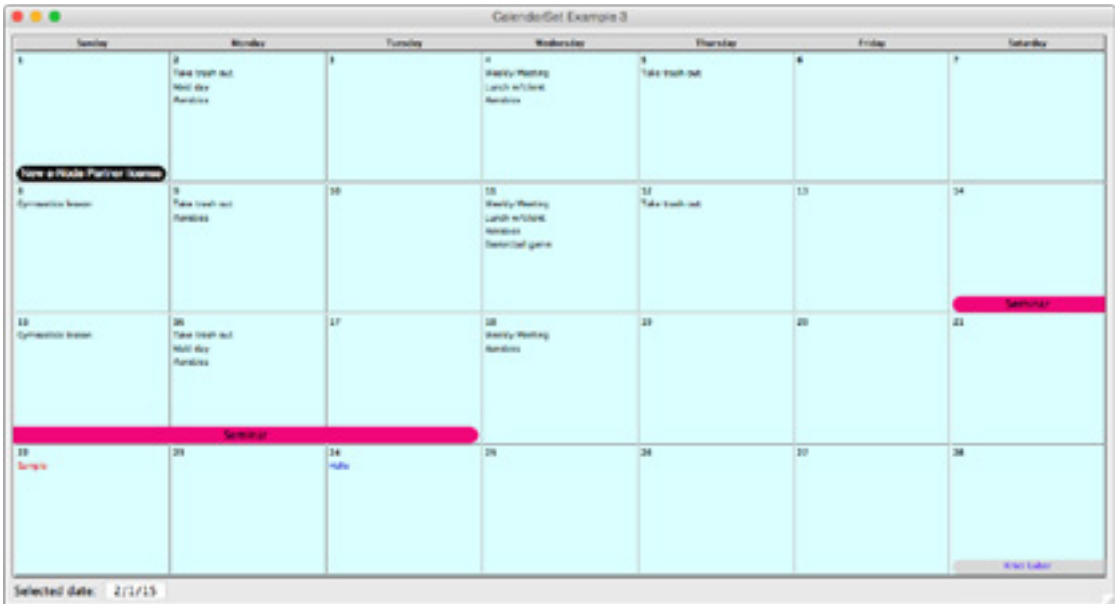
CS_SetEventOpts is not used, therefore event selection is disabled (default).

Conversely, CS_SetBanrOpts is not used either but this function defaults its parameters to 1 in this case (AllowBannerSelect, AllowBannerResize): banners are selectable and resizable.

Our new object method is:

```
Case of
   : (Form event=On Load)
      vSelDate:=!00/00/00!
       // set the range of the calendar to be Feb 2015
      CS_SetRange(Self->;!02/01/2015!;!02/28/2015!)
       // Items of type 1 are text items
       // Find all of the text items
      QUERY([Cal Items];[Cal Items]ItemType=1)
       // load the arrays with the data
      SELECTION TO ARRAY([Cal Items]StartDate;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(Self->;"vTextDate";"vTextDesc";"vTextFont";"vTextSize";"vTextStyle";"vTextColor")
       // Items of type 2 are Banner items
       // Find all of the Banner items
      QUERY([Cal Items];[Cal Items]ItemType=2)
       // pass the arrays to CalendarSet
      SELECTION TO ARRAY([Cal Items]StartDate;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(Self->;"vBanrDateS";"vBanrDateE";"vBanrDesc";"vBanrFont";\
         "vBanrSize";"vBanrStyle";"vBanrColorF";"vBanrColorB")

   :(Form event=On Clicked)
      CS_GetAction (Self->;vAction)
      Case of
         :(vAction=CS_Action_ClickOnDay)  // clicked on a day
            CS_GetSelect(Self->;vSelDate;vSelDateE;isContig;aDates)
         :(vAction=CS_Action_ClickOnEventBanner)|(vAction=CS_Action_BannerResize)
          // clicked on a banner (including resize) - events default to non selectable so this is a banner
            CS_GetEvtSelect (Self->;vEvtType;vIndex)
            vSelDate:= vBanrDateS{vIndex}  // start date of this banner
      End case
End case
```

Our form now looks like this to the user:



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 a vSelected text variable which shows information about the day or days selected.

Here's the modified object method:

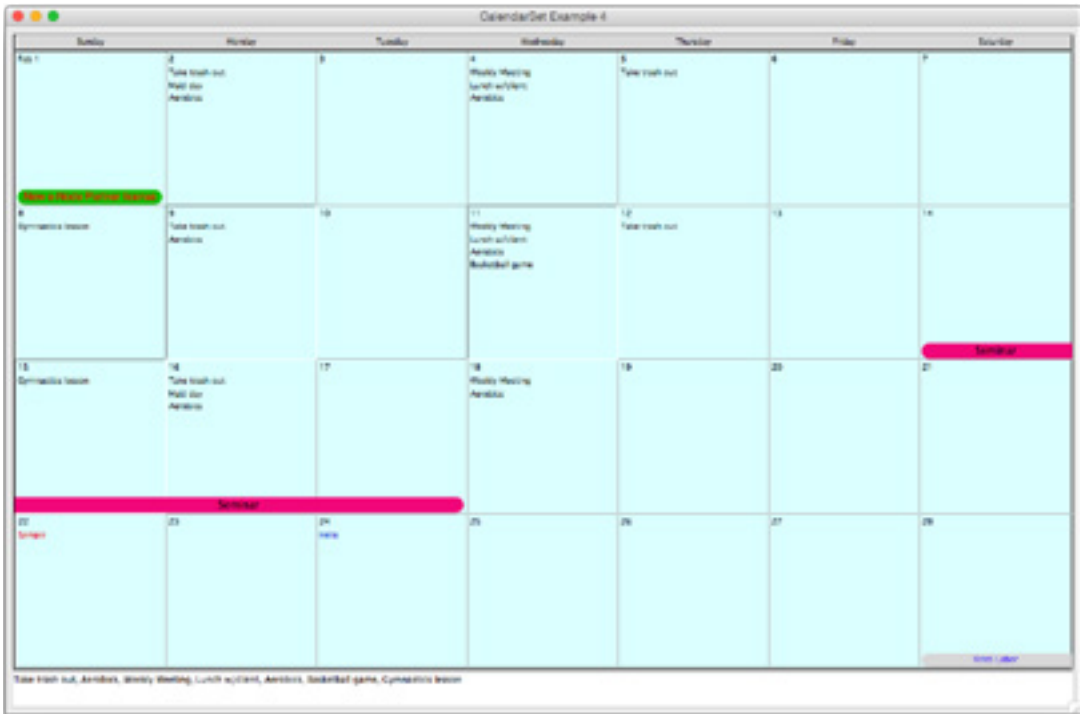
```
Case of
   : (Form event=On Load)
      vSelDate:=!00/00/0000!
      vSelDateE:=!00/00/0000!
      vSelected:="Select one or several day(s) or a banner..."
      ARRAY LONGINT(vSelItems;0)  // index of selected items
      ARRAY TEXT(vSelText;0)  // selected items text
       // set the range of the calendar to be Feb 2015
      CS_SetRange(Self->;!02/01/15!;!02/28/15!)
       // Items of type 1 are text items
       // Find all of the text items
      QUERY([Cal Items];[Cal Items]ItemType=1)
       // load the arrays with the data
      SELECTION TO ARRAY([Cal Items]StartDate;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 (Self->;"vTextDate";"vTextDesc";"vTextFont";"vTextSize";"vTextStyle";"vTextColor")
       // Items of type 2 are Banner items
       // Find all of the Banner items
      QUERY([Cal Items];[Cal Items]ItemType=2)  // load the arrays with the data
      SELECTION TO ARRAY([Cal Items]StartDate;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(Self->;"vBanrDateS";"vBanrDateE";\
         "vBanrDesc";"vBanrFont";"vBanrSize";"vBanrStyle";"vBanrColorF";"vBanrColorB")
       // Set the options to display entire highlight, allow discontiguous selection,
       // do not show day number for unused days, display month names and OS first day of the week
      CS_Options (Self->;CS_Highlight_Cell;CS_DaySelect_Discontiguous;\
         CS_Unused_GrayedEmpty;CS_MonthOnFirstOn;CS_FirstDayOS)
   : (Form event=On Clicked)
      CS_GetAction (Self->;vAction)
      Case of
         :(vAction=CS_Action_ClickOnDay)  // clicked on a day
            CS_GetSelect(Self->;vSelDate;vSelDateE;isContig;aDates)
             // get an index of the selected items
            CS_GetSelItems (Self->;CS_Sel_Events;vSelItems)
             // create an array of the selected items only, using the index
            FS_ArrayIntrsct (vSelItems;vTextDesc;vSelText)
            vSelected:=ArrayToText (->vSelText;", ")
```

```
:(vAction=CS_Action_ClickOnEventBanner)|(vAction=CS_Action_BannerResize)
   //clicked on a banner (including resize) - events default to non selectable so this is a banner
      CS_GetEvtSelect (Self->;vEvtType;vIndex)
      vSelected:="Banner: "+vBanrDesc{vIndex}  //text associated to this banner
End case
End case
```

The 4D ***ArrayToText*** project method simply concatenates text array items into a text variable:

```
C_POINTER($1)  //->array
C_TEXT($2)  //separator
C_TEXT($0)  //text
C_LONGINT($i)
$0:=""
For ($i;1;Size of array($1->))
   $0:=$0+$1->{$i}+$2
End for
$0:=Substring($0;1;Length($0)-Length($2))  //remove last separator
```

Here is the resulting form:



Example 5 — Dragging Events within a CalendarSet Object

Once again we'll modify the example, to allow events and banners 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).

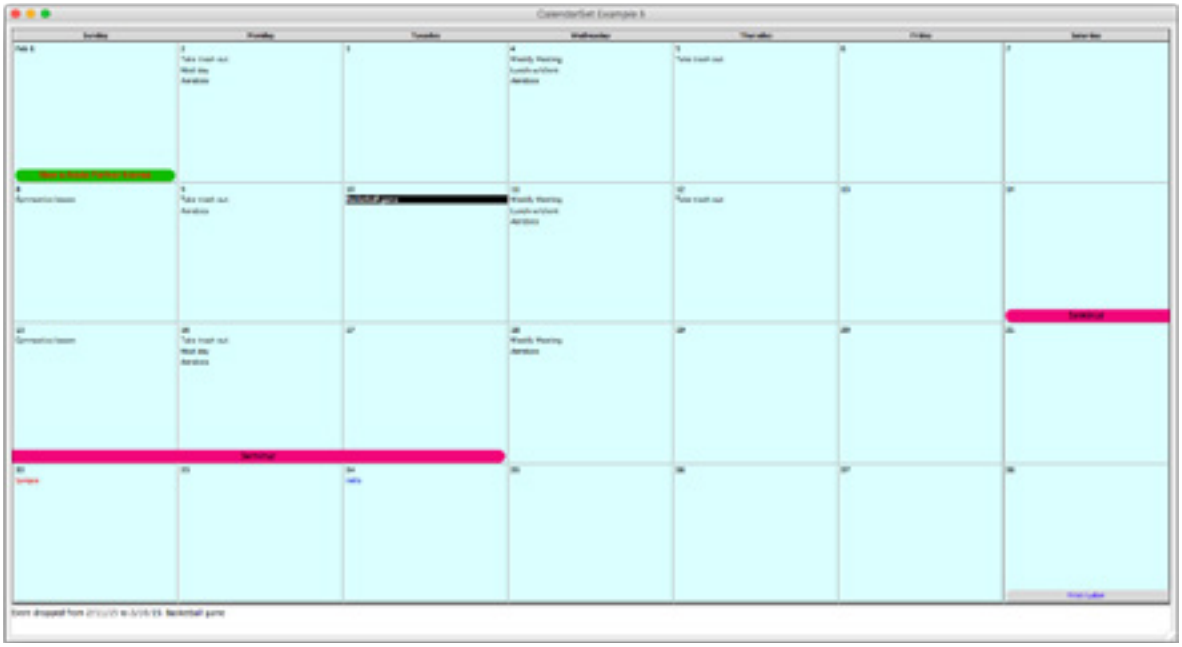
We use CS_SetDrgSrc and CS_SetDrgDst to restrict event dragging to be within the object.

Here's the modified object method:

```
Case of
   : (Form event=On Load)
      vSelected:="Drag events or banners..."
       // set the range of the calendar to be Feb 2015
      CS_SetRange(Self->;!02/01/15!;!02/28/15!)
       // Items of type 1 are text items
       // Find all of the text items
      QUERY([Cal Items];[Cal Items]ItemType=1)
       // load the arrays with the data
      SELECTION TO ARRAY([Cal Items]StartDate;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 (Self->;"vTextDate";"vTextDesc";"vTextFont";"vTextSize";"vTextStyle";"vTextColor")
       // Items of type 2 are Banner items
       // Find all of the Banner items
      QUERY([Cal Items];[Cal Items]ItemType=2)
       // load the arrays with the data
      SELECTION TO ARRAY([Cal Items]StartDate;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 (Self->;"vBanrDateS";"vBanrDateE";"vBanrDesc"; \
         "vBanrFont";"vBanrSize";"vBanrStyle";"vBanrColorF";"vBanrColorB")
       // Set the options to display entire highlight, allow discontiguous selection,
       // do not show day number for unused days, display month names and OS first day of the week
      CS_Options (Self->;CS_Highlight_Cell;CS_DaySelect_Discontiguous;\
         CS_Unused_GrayedEmpty;CS_MonthOnFirstOn;CS_FirstDayOS)
       // enable dragging banners and events to days within this area
      vAccessCode:="CalendarSetArea"+String(Self->)  // allows dragging within this area
       //make events selectable to enable dragging (see the note to CS_SetDrgSrc)
      CS_SetEventOpts(Self->;CS_EventSelect_Single)
      CS_SetDrgSrc (Self->;CS_DragSource_EventBanner;vAccessCode)  // source: event or banner
      CS_SetDrgDst (Self->;CS_DragDestination_Day;vAccessCode)  // destination: day
```

```
: (Form event=On Drop)  // could also be CS_Action_Drop from CS_GetAction
   C_LONGINT($SourceArea;$SourceProcessID)
   C_DATE($StartDate)
   CS_GetDrgSrcArea (Self->;$SourceArea;$SourceProcessID)  // or DRAG AND DROP PROPERTIES
   If ($SourceArea=Self->)  // drag within
      CS_GetDrgSrcEvt (Self->;vDataType;vEventIndex;$StartDate)  // what was dragged
      CS_GetDrgDstDay (Self->;$DestDate)  // to which day
      Case of
      : ($DestDate=!00/00/0000!)  // invalid date ("unused day")
      : (vDataType=CS_Type_Event)  // only one event can be selected at a time (CS_EventSelect_Single)
         vSelected:="Event dropped from "+String($StartDate)+" to "+String($DestDate)+": "\
            +vTextDesc{vEventIndex}
      : (vDataType=CS_Type_Banner)  // only one banner can be selected at a time
         vSelected:="Banner dropped from "+String($StartDate)+" to "+String($DestDate)+": "\
            +vBanrDesc{vEventIndex}
      End case
   End if
End case
```

Here is the resulting form:



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

Our previous example only allowed events and banners 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 form. The two objects are named eCalendar and eOther.

Instead of displaying a message as in the previous example, we will update both calendars once a drop has occurred. When the user drags and drop an event from eCalendar to eOther, we'll remove that event from eCalendar and "paste" it into the arrays displayed in eOther.

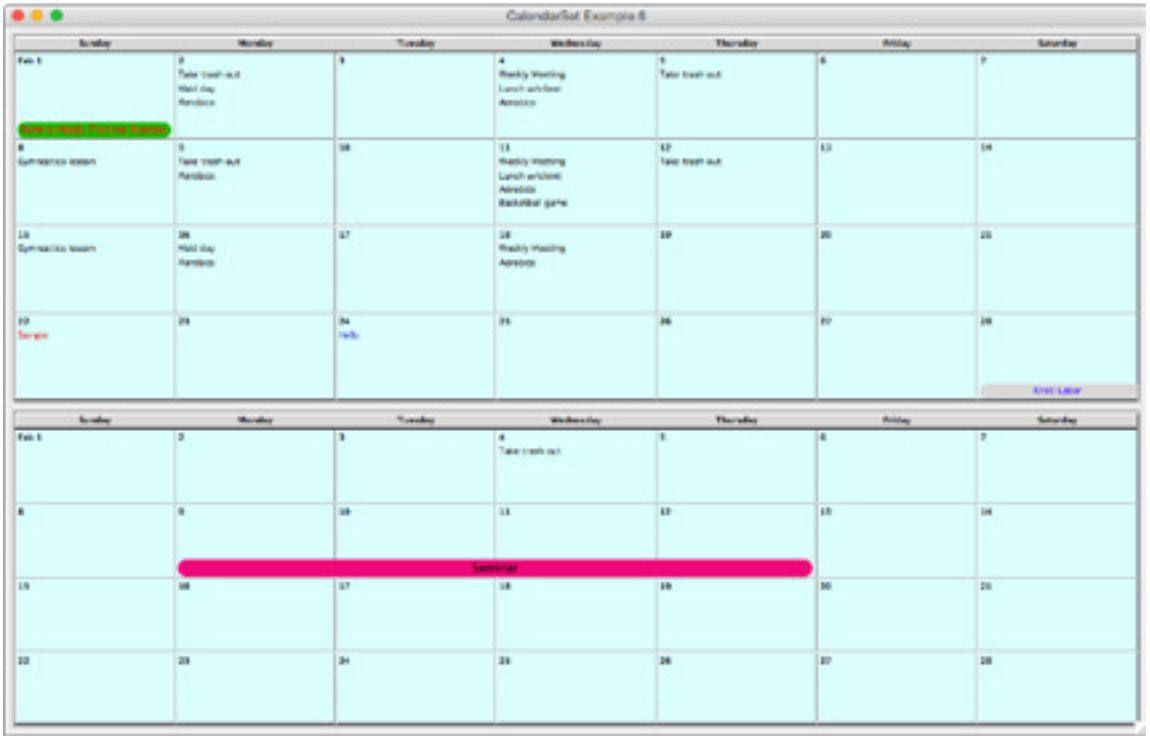
We'll show both the object method for eCalendar (loading and displaying data) and the destination area eOther (setting an empty calendar on load, then updating both areas after a drop).

```
//eCalendar (source area) object method
Case of
   : (Form event=On Load)
      //set the range of the calendar to be Feb 2015
      CS_SetRange(Self->;!02/01/15!;!02/28/15!)
      //Items of type 1 are text items
      //Find all of the text items
      QUERY([Cal Items];[Cal Items]ItemType=1)
      //load the arrays with the data
      SELECTION TO ARRAY([Cal Items]StartDate;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 (Self->;"vTextDate";"vTextDesc";"vTextFont";"vTextSize";"vTextStyle";"vTextColor")
      //Items of type 2 are Banner items
      //Find all of the Banner items
      QUERY([Cal Items];[Cal Items]ItemType=2)
      //load the arrays with the data
      SELECTION TO ARRAY([Cal Items]StartDate;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 (Self->;"vBanrDateS";"vBanrDateE";"vBanrDesc"; \
         "vBanrFont";"vBanrSize";"vBanrStyle";"vBanrColorF";"vBanrColorB")
      //Set the options to display entire highlight, allow discontiguous selection,
      //do not show day number for unused days, display month names and OS first day of the week
      CS_Options (Self->;CS_Highlight_Cell;CS_DaySelect_Discontiguous;\
         CS_Unused_GrayedEmpty;CS_MonthOnFirstOn;CS_FirstDayOS)
      //enable dragging events and banners from this area to days in eOther
      vAccessCode:="CalendarSetArea"+String(Self->)+"to"+String(eOther)  //allows dragging between
      CS_SetEventOpts(Self->;CS_EventSelect_Single)  //make events selectable to enable dragging
      CS_SetDrgSrc (Self->;CS_DragSource_EventBanner;vAccessCode)  //source: event or banner
      CS_SetDrgDst (eOther;CS_DragDestination_Day;vAccessCode)  //destination: day in eOther
End case
```

```
 //eOther (destination area) object method
Case of
   : (Form event=On Load)
      //set the range of the calendar to be Feb 2015
      CS_SetRange(Self->;!02/01/15!;!02/28/15!)
      //Empty arrays for events and banners
      ARRAY DATE(vTextDateDes;0)
      ARRAY TEXT(vTextDescDes;0)
      ARRAY TEXT(vTextFontDes;0)
      ARRAY INTEGER(vTextSizeDes;0)
      ARRAY INTEGER(vTextStyleDes;0)
      ARRAY INTEGER(vTextColorDes;0)
      CS_SetArray (Self->;"vTextDateDes";"vTextDescDes";"vTextFontDes";\
         "vTextSizeDes";"vTextStyleDes";"vTextColorDes")
      ARRAY DATE(vBanrDateSDes;0)
      ARRAY DATE(vBanrDateEDes;0)
      ARRAY TEXT(vBanrDescDes;0)
      ARRAY TEXT(vBanrFontDes;0)
      ARRAY INTEGER(vBanrSizeDes;0)
      ARRAY INTEGER(vBanrStyleDes;0)
      ARRAY INTEGER(vBanrColorFDes;0)
      ARRAY INTEGER(vBanrColorBDes;0)
      //Set the options to display entire highlight, allow discontiguous selection,
      //do not show day number for unused days, display month names and OS first day of the week
      CS_SetBanrArray (Self->;"vBanrDateSDes";"vBanrDateEDes";"vBanrDescDes";\
         "vBanrFontDes";"vBanrSizeDes";"vBanrStyleDes";"vBanrColorFDes";"vBanrColorBDes")
   : (Form event=On Drop)  //could also be CS_Action_Drop from CS_GetAction
      C_LONGINT($SourceArea;$SourceProcessID)
      C_DATE($StartDate)
      CS_GetDrgSrcArea (Self->;$SourceArea;$SourceProcessID)  //or DRAG AND DROP PROPERTIES
      If ($SourceArea=eCalendar)  //drag from eCalendar
         CS_GetDrgSrcEvt (Self->;vDataType;vEventIndex;$StartDate)  //what was dragged
         CS_GetDrgDstDay (Self->;$DestDate)  //to which day
         Case of  //we check that the destination date is within the range set with CS_SetRange
         : ($DestDate=!00/00/0000!)  //invalid date ("unused day")
         : (vDataType=CS_Type_Event)  //only one event can be selected at a time (CS_EventSelect_Single)
            //Add to destination
            APPEND TO ARRAY(vTextDateDes;$DestDate)
            APPEND TO ARRAY(vTextDescDes;vTextDesc{vEventIndex})
            APPEND TO ARRAY(vTextFontDes;vTextFont{vEventIndex})
            APPEND TO ARRAY(vTextSizeDes;vTextSize{vEventIndex})
            APPEND TO ARRAY(vTextStyleDes;vTextStyle{vEventIndex})
            APPEND TO ARRAY(vTextColorDes;vTextColor{vEventIndex})
```

```
            //Remove from source
            DELETE FROM ARRAY(vTextDate;vEventIndex)
            DELETE FROM ARRAY(vTextDesc;vEventIndex)
            DELETE FROM ARRAY(vTextFont;vEventIndex)
            DELETE FROM ARRAY(vTextSize;vEventIndex)
            DELETE FROM ARRAY(vTextStyle;vEventIndex)
            DELETE FROM ARRAY(vTextColor;vEventIndex)
         : (vDataType=CS_Type_Banner)  //only one banner can be selected at a time (default)
            //Add to destination
            APPEND TO ARRAY(vBanrDateSDes;$DestDate)
            APPEND TO ARRAY(vBanrDateEDes;$DestDate+\
               (vBanrDateE{vEventIndex}-vBanrDateS{vEventIndex}))  //same duration
            APPEND TO ARRAY(vBanrDescDes;vBanrDesc{vEventIndex})
            APPEND TO ARRAY(vBanrFontDes;vBanrFont{vEventIndex})
            APPEND TO ARRAY(vBanrSizeDes;vBanrSize{vEventIndex})
            APPEND TO ARRAY(vBanrStyleDes;vBanrStyle{vEventIndex})
            APPEND TO ARRAY(vBanrColorFDes;vBanrColorF{vEventIndex})
            APPEND TO ARRAY(vBanrColorBDes;vBanrColorB{vEventIndex})
            //Remove from source
            DELETE FROM ARRAY(vBanrDateS;vEventIndex)
            DELETE FROM ARRAY(vBanrDateE;vEventIndex)
            DELETE FROM ARRAY(vBanrDesc;vEventIndex)
            DELETE FROM ARRAY(vBanrFont;vEventIndex)
            DELETE FROM ARRAY(vBanrSize;vEventIndex)
            DELETE FROM ARRAY(vBanrStyle;vEventIndex)
            DELETE FROM ARRAY(vBanrColorF;vEventIndex)
            DELETE FROM ARRAY(vBanrColorB;vEventIndex)
         End case
         Area_Refresh (eCalendar)
         Area_Refresh (Self->)
      End if
   End case
```

Here is the resulting form, after dropping a banner and an event:



Example 7 — Dragging Events to a CalendarSet Object on a Different Form

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

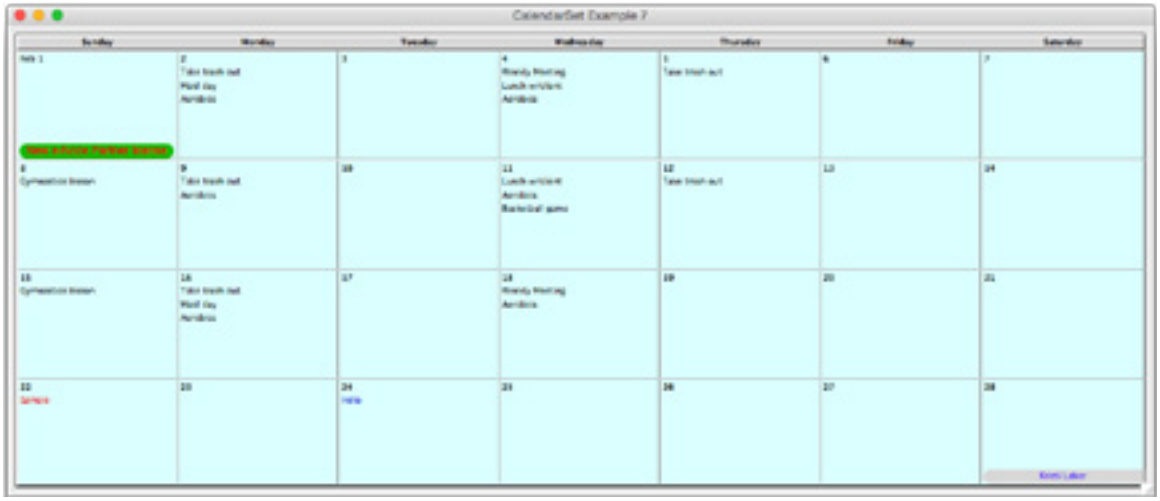
We have to use the 4D process communication commands to assist with the implementation. When we detect a drop from eCalendar to eOther, our eOther object method will update the arrays.

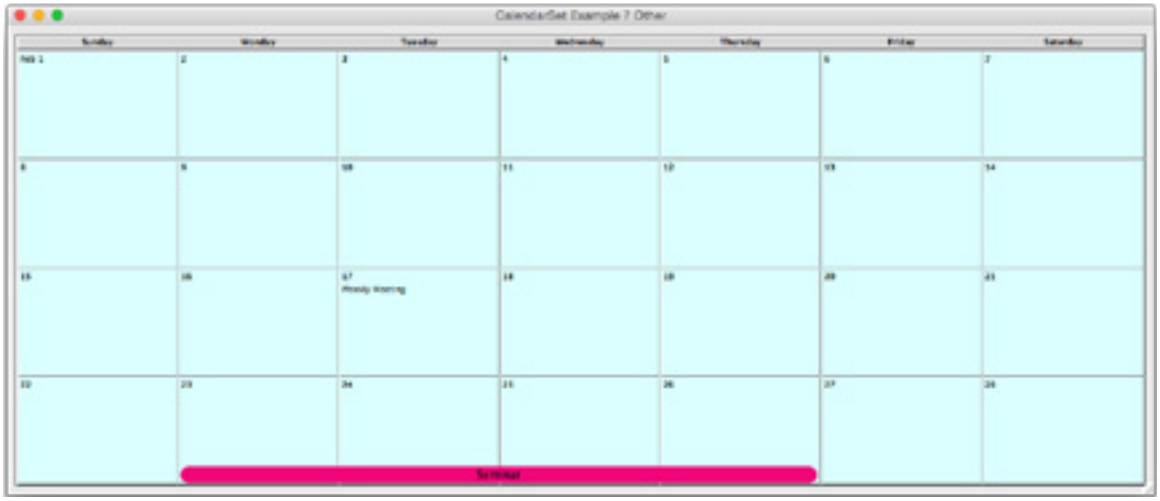
```
// eCalendar (source area) object method
Case of
   : (Form event=On Load)
      // we'll not show the standard setup stuff, and just show the dragging-specific code
      (...)
      // store both process numbers in interprocess variables for process communication
      <>DragDemo:=Current process
      // display the other form in its own process
      <>DragDemo2:=New process("Example";0;"Other process";"7 Other")
      // enable dragging events and banners from this area to days in eOther
      vAccessCode:="CalendarSetArea"+String(<>DragDemo)+"to"+String(<>DragDemo2)
      CS_SetEventOpts (Self->;CS_EventSelect_Single)  // make events selectable to enable dragging
      CS_SetDrgSrc (Self->;CS_DragSource_EventBanner;vAccessCode)  // source: event or banner
      // destination is set in the other process (eOther object method)
End case


// eOther object method
Case of
   : (Form event=On Load)
      // we'll not show the standard setup stuff, and just show the dragging-specific code
      (...)
      vAccessCode:="CalendarSetArea"+String(<>DragDemo)+"to"+String(<>DragDemo2)
      CS_SetDrgDst(eOther;CS_DragDestination_Day;vAccessCode)  // destination: day
   : (Form event=On Drop)  // could also be CS_Action_Drop from CS_GetAction
      C_LONGINT($SourceArea;$SourceProcessID)
      C_DATE($StartDate)
      CS_GetDrgSrcArea (Self->;$SourceArea;$SourceProcessID)  // or DRAG AND DROP PROPERTIES
      If ($SourceProcessID=<>DragDemo)  // drag from eCalendar in the source process
         CS_GetDrgSrcEvt (Self->;vDataType;vEventIndex;$StartDate)  // what was dragged
         CS_GetDrgDstDay (Self->;$DestDate)  // to which day
         If ($DestDate#!00/00/0000!)  // valid date (not "unused day")
            C_LONGINT($Size)  // new array size
            Case of
               : (vDataType=CS_Type_Event)
                  // only one event can be selected at a time (CS_EventSelect_Single)
                     // Add to destination
                  APPEND TO ARRAY(vTextDateDes;$DestDate)
                  $Size:=Size of array(vTextDateDes)
```

```
            INSERT IN ARRAY(vTextDescDes;$Size;1)
            GET PROCESS VARIABLE(<>DragDemo;vTextDesc{vEventIndex};vTextDescDes{$Size})
            INSERT IN ARRAY(vTextFontDes;$Size;1)
            GET PROCESS VARIABLE(<>DragDemo;vTextFont{vEventIndex};vTextFontDes{$Size})
            INSERT IN ARRAY(vTextSizeDes;$Size;1)
            GET PROCESS VARIABLE(<>DragDemo;vTextSize{vEventIndex};vTextSizeDes{$Size})
            INSERT IN ARRAY(vTextStyleDes;$Size;1)
            GET PROCESS VARIABLE(<>DragDemo;vTextStyle{vEventIndex};vTextStyleDes{$Size})
            INSERT IN ARRAY(vTextColorDes;$Size;1)
            GET PROCESS VARIABLE(<>DragDemo;vTextColor{vEventIndex};vTextColorDes{$Size})
         : (vDataType=CS_Type_Banner)  //only one banner can be selected at a time (default)
            //Add to destination
            $Size:=Size of array(vBanrDateEDes)+1
            INSERT IN ARRAY(vBanrDateSDes;$Size;1)
            //temporary report (source start date)
            GET PROCESS VARIABLE(<>DragDemo;vBanrDateS{vEventIndex};vBanrDateSDes{$Size})
            INSERT IN ARRAY(vBanrDateEDes;$Size;1)
            //temporary report (source end date)
            GET PROCESS VARIABLE(<>DragDemo;vBanrDateE{vEventIndex};vBanrDateEDes{$Size})
            //Real dates according to the original duration
            vBanrDateEDes{$Size}:=$DestDate+(vBanrDateEDes{$Size}-vBanrDateSDes{$Size})
            vBanrDateSDes{$Size}:=$DestDate
            INSERT IN ARRAY(vBanrDescDes;$Size;1)
            GET PROCESS VARIABLE(<>DragDemo;vBanrDesc{vEventIndex};vBanrDescDes{$Size})
            INSERT IN ARRAY(vBanrFontDes;$Size;1)
            GET PROCESS VARIABLE(<>DragDemo;vBanrFont{vEventIndex};vBanrFontDes{$Size})
            INSERT IN ARRAY(vBanrSizeDes;$Size;1)
            GET PROCESS VARIABLE(<>DragDemo;vBanrSize{vEventIndex};vBanrSizeDes{$Size})
            INSERT IN ARRAY(vBanrStyleDes;$Size;1)
            GET PROCESS VARIABLE(<>DragDemo;vBanrStyle{vEventIndex};vBanrStyleDes{$Size})
            INSERT IN ARRAY(vBanrColorFDes;$Size;1)
            GET PROCESS VARIABLE(<>DragDemo;vBanrColorF{vEventIndex};vBanrColorFDes{$Size})
            INSERT IN ARRAY(vBanrColorBDes;$Size;1)
            GET PROCESS VARIABLE(<>DragDemo;vBanrColorB{vEventIndex};vBanrColorBDes{$Size})
      End case
      //Inform source: item to remove (index, type)
      VARIABLE TO VARIABLE(<>DragDemo;vEventIndex;vEventIndex;vDataType;vDataType)
      CALL PROCESS(<>DragDemo)
      Area_Refresh (Self->)
      End if  //valid date
   End if  //$SourceProcessID=<>DragDemo
End case
```

```
 //Destination form method (On Outside Call event)
 //eOther is calling: remove dragged item
Case of  //playing safe
   : (vEventIndex<1)
   : (vEventIndex>Size of array(vTextDate))
   : (vDataType=CS_Type_Event)
      DELETE FROM ARRAY(vTextDate;vEventIndex)
      DELETE FROM ARRAY(vTextDesc;vEventIndex)
      DELETE FROM ARRAY(vTextFont;vEventIndex)
      DELETE FROM ARRAY(vTextSize;vEventIndex)
      DELETE FROM ARRAY(vTextStyle;vEventIndex)
      DELETE FROM ARRAY(vTextColor;vEventIndex)
   : (vDataType=CS_Type_Banner)
      DELETE FROM ARRAY(vBanrDateS;vEventIndex)
      DELETE FROM ARRAY(vBanrDateE;vEventIndex)
      DELETE FROM ARRAY(vBanrDesc;vEventIndex)
      DELETE FROM ARRAY(vBanrFont;vEventIndex)
      DELETE FROM ARRAY(vBanrSize;vEventIndex)
      DELETE FROM ARRAY(vBanrStyle;vEventIndex)
      DELETE FROM ARRAY(vBanrColorF;vEventIndex)
      DELETE FROM ARRAY(vBanrColorB;vEventIndex)
End case
Area_Refresh (eCalendar)
```

Here are the resulting forms, after dropping a banner and an event:





Example 8 — Dragging from AreaList Pro to CalendarSet

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

In this example we will also display event times.

We use the vEventIndex variable as a communication “token” between the two processes:

- When a drop is received, destination (CalendarSet) process sends vEventIndex=-1 to source (AreaList Pro) process, meaning “send me the index number of the row that was dragged”.
- Source process assigns the index value to vEventIndex for destination process to read.
- Destination process updates the CalendarSet area, then sends vEventIndex=-2 to source process, meaning “delete the selected row”.

This is one of many tricks that can be used for this kind of process communication.

First, the object method of the AreaList Pro (source) area (On Load event):

```
 //load events
QUERY([Cal Items];[Cal Items]ItemType=1)
SELECTION TO ARRAY([Cal Items]Description;vTextDesc;[Cal Items]Time;vTextTime;\
   [Cal Items]Font;vTextFont; [Cal Items]Size;vTextSize;[Cal Items]Style;vTextStyle;\
   [Cal Items]ForeColor;vTextColor)

 //column setup (array mode)
C_LONGINT($error)
$error:=AL_AddColumn (Self->;->vTextDesc)
$error:=AL_AddColumn (Self->;->vTextTime)
$error:=AL_AddColumn (Self->;->vTextFont)
$error:=AL_AddColumn (Self->;->vTextSize)
$error:=AL_AddColumn (Self->;->vTextStyle)
$error:=AL_AddColumn (Self->;->vTextColor)

 //column headers
AL_SetColumnTextProperty (Self->;1;ALP_Column_HeaderText;"Description")
AL_SetColumnTextProperty (Self->;2;ALP_Column_HeaderText;"Time")
AL_SetColumnTextProperty (Self->;3;ALP_Column_HeaderText;"Font")
AL_SetColumnTextProperty (Self->;4;ALP_Column_HeaderText;"Size")
AL_SetColumnTextProperty (Self->;5;ALP_Column_HeaderText;"Style")
AL_SetColumnTextProperty (Self->;6;ALP_Column_HeaderText;"Color")

 //formatting
AL_SetColumnTextProperty (Self->;2;ALP_Column_Format;"&/105")
AL_SetColumnTextProperty (Self->;4;ALP_Column_Format;"##0")
AL_SetColumnTextProperty (Self->;5;ALP_Column_Format;"##0")
AL_SetColumnTextProperty (Self->;6;ALP_Column_Format;"##0")

 //make it nicer
AL_SetAreaLongProperty (Self->;ALP_Area_AutoResizeColumn;1)  //autoresize columns 1
AL_SetAreaLongProperty (Self->;ALP_Area_AltRowOptions;1)  //alternate row color
AL_SetAreaLongProperty (Self->;ALP_Area_RowIndentH;10)  //row indent (horizontal padding)
```

```
  // store both process numbers in interprocess variables for process communication
<>DragDemo:=Current process
  // display the other form in its own process
<>DragDemo2:=New process("Example";0;"Other process";"8 Other")
  // enable dragging rows from this AreaList Pro area to days in eOther CalendarSet area
vAccessCode:="AreaListProArea"+String(<>DragDemo)+"to"+String(<>DragDemo2)
AL_SetAreaTextProperty (Self->;ALP_Area_DragSrcRowCodes;vAccessCode)
  // area has been made draggable
  // destination (CalendarSet area) is set in the other process (eOther object method)
```

Next, the object method of the CalendarSet object in the destination form:

```
Case of
   : (Form event=On Load)
      GET PROCESS VARIABLE(<>DragDemo;eALP;eALP)  // source area reference
       // set the range of the calendar to be Feb 2015
      CS_SetRange(Self->;!02/01/15!;!02/28/15!)
       // Empty arrays for events
      ARRAY DATE(vTextDateDes;0)
      ARRAY LONGINT(vTextTimeDes;0)
      ARRAY TEXT(vTextDescDes;0)
      ARRAY TEXT(vTextFontDes;0)
      ARRAY INTEGER(vTextSizeDes;0)
      ARRAY INTEGER(vTextStyleDes;0)
      ARRAY INTEGER(vTextColorDes;0)
      CS_SetArray (Self->;"vTextDateDes";"vTextDescDes";"vTextFontDes";\
         "vTextSizeDes";"vTextStyleDes";"vTextColorDes";"vTextTimeDes")
       //  show times
      CS_SetEventOpts(Self->CS_EventSelect_Single;CS_WordWrap_Off;CS_Marker_None;CS_EventTime_Show)
       // Set the options to display entire highlight, allow discontiguous selection,
       // do not show day number for unused days, display month names and OS first day of the week
      CS_Options (Self->;CS_Highlight_Cell;CS_DaySelect_Discontiguous;\
         CS_Unused_GrayedEmpty;CS_MonthOnFirstOn;CS_FirstDayOS)
       // enable dragging events to days in this area from the source AreaList Pro area
      vAccessCode:="AreaListProArea"+String(<>DragDemo->)+"to"+String(<>DragDemo2)
      CS_SetDrgDst (Self->;CS_DragDestination_Day;vAccessCode)  // destination: day
   : (Form event=On Drop)  // could also be CS_Action_Drop from CS_GetAction
      C_LONGINT($SourceArea;$SourceProcessID)
      C_DATE($StartDate)
      CS_GetDrgSrcArea (Self->;$SourceArea;$SourceProcessID)  // or DRAG AND DROP PROPERTIES
      If ($SourceArea=-eALP)  // drag from eALP in the source process (value is negative)
         CS_GetDrgDstDay (Self->;$DestDate)  // to which day
         If ($DestDate#!00/00/0000!)  // valid date (not "unused day")
```

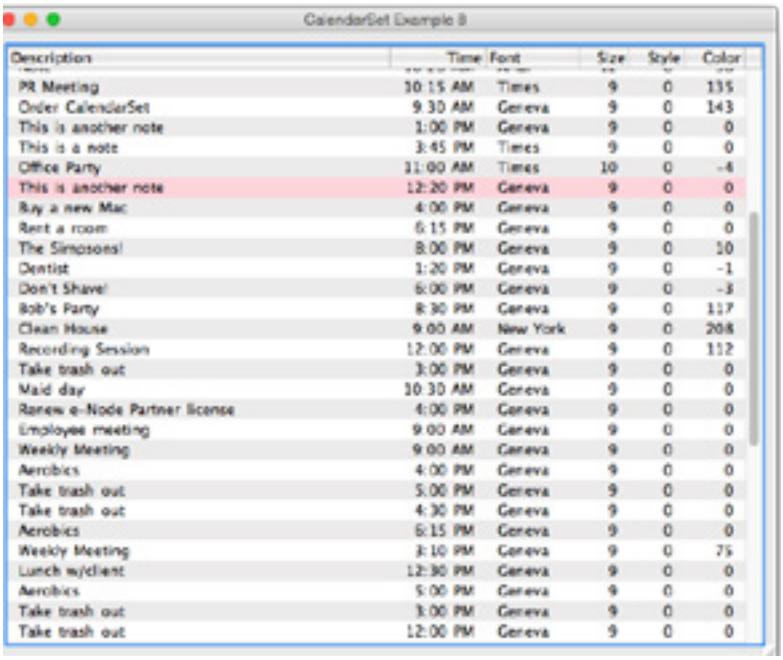
```
            // Get information from source
            vEventIndex:=-1  // ask eALP process for selected row
            VARIABLE TO VARIABLE(<>DragDemo;vEventIndex;vEventIndex)
            CALL PROCESS(<>DragDemo)
            While (vEventIndex=-1)  // wait for returned value
               GET PROCESS VARIABLE(<>DragDemo;vEventIndex;vEventIndex)  // index of selected row
            End while
            C_LONGINT($Size)  // new array size
            APPEND TO ARRAY(vTextDateDes;$DestDate)
            $Size:=Size of array(vTextDateDes)
            INSERT IN ARRAY(vTextDescDes;$Size;1)
            GET PROCESS VARIABLE(<>DragDemo;vTextDesc{vEventIndex};vTextDescDes{$Size})
            INSERT IN ARRAY(vTextTimeDes;$Size;1)
            GET PROCESS VARIABLE(<>DragDemo;vTextTime{vEventIndex};vTextTimeDes{$Size})
            INSERT IN ARRAY(vTextFontDes;$Size;1)
            GET PROCESS VARIABLE(<>DragDemo;vTextFont{vEventIndex};vTextFontDes{$Size})
            INSERT IN ARRAY(vTextSizeDes;$Size;1)
            GET PROCESS VARIABLE(<>DragDemo;vTextSize{vEventIndex};vTextSizeDes{$Size})
            INSERT IN ARRAY(vTextStyleDes;$Size;1)
            GET PROCESS VARIABLE(<>DragDemo;vTextStyle{vEventIndex};vTextStyleDes{$Size})
            INSERT IN ARRAY(vTextColorDes;$Size;1)
            GET PROCESS VARIABLE(<>DragDemo;vTextColor{vEventIndex};vTextColorDes{$Size})
         // Inform source: item to remove
            vEventIndex:=-2  // ask eALP process to delete selected row
         VARIABLE TO VARIABLE(<>DragDemo;vEventIndex;vEventIndex)
         CALL PROCESS(<>DragDemo)
         Area_Refresh (Self->)
         End if  // valid date
      End if  // $SourceProcessID=-eALP
   End case
```

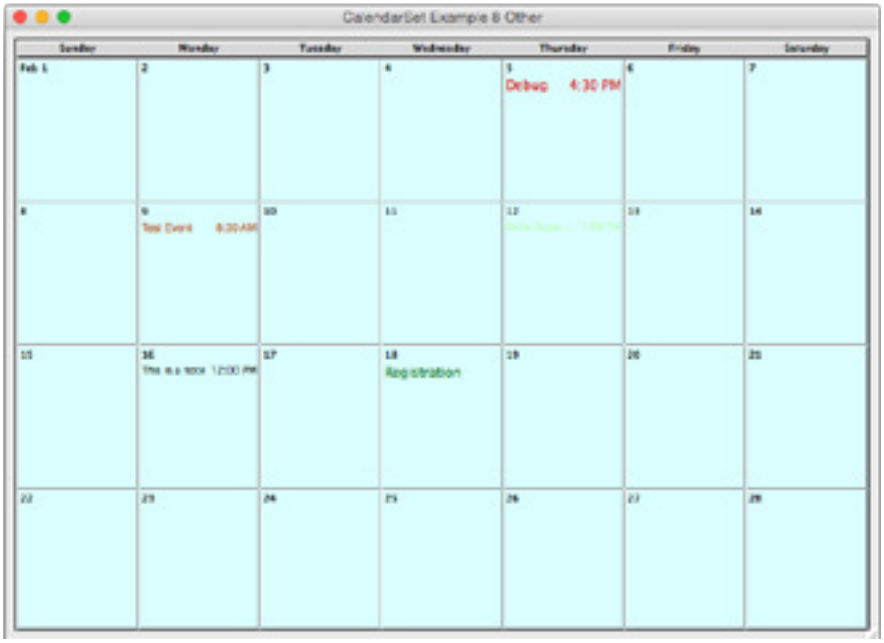
The process communication stuff is managed by the source form method (Outside Call event) in response to the destination area calls:

```
   Case of
      : (vEventIndex=-1)  // after a drag and drop: get selected row
         vEventIndex:=AL_GetAreaLongProperty (eALP;ALP_Area_SelRow)  // selected row
      : (vEventIndex=-2)  // delete row
         vEventIndex:=AL_GetAreaLongProperty (eALP;ALP_Area_SelRow)  // selected row
         Case of   // playing safe
            : (vEventIndex<1)
            : (vEventIndex>Size of array(vTextDesc))
         Else
            DELETE FROM ARRAY(vTextDesc;vEventIndex)
            DELETE FROM ARRAY(vTextTime;vEventIndex)
            DELETE FROM ARRAY(vTextFont;vEventIndex)
```

```
            DELETE FROM ARRAY(vTextSize;vEventIndex)
            DELETE FROM ARRAY(vTextStyle;vEventIndex)
            DELETE FROM ARRAY(vTextColor;vEventIndex)
        End case
    AL_SetAreaLongProperty (eALP;ALP_Area_UpdateData;0)  // refresh area
End case
```

Here are the resulting forms, after dropping a few events:





Index

A

B

C

D

E

F

I

L

M

O

P

R

S

T

U

V

W

X

Copyrights and Trademarks

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

CalendarSet is copyright Plugin Masters SAS and exclusively published worldwide by e-Node.

4th Dimension, 4D and 4D Server are trademarks of 4D SAS.

Windows, Excel and Vista are trademarks of Microsoft Corporation.

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