

# SchedulePack™

Version 4.0



by Soft Solutions



Results Through Focused Technology...  
*1998-2010 Soft Solutions, Inc. All Rights Reserved.*  
Published World-Wide by Soft Solutions, Inc.

**Soft Solutions, Inc.**  
2900 Chamblee Tucker Road  
Building 12, Suite 200  
Atlanta, GA 30341 USA

**IMPORTANT—READ CAREFULLY BEFORE OPENING.** By installing this software plug-in, you indicate your acceptance of the following Soft Solutions, Inc. License Agreement.

### **Software License and Limited Warranty Agreement:**

This is a legal agreement between you, the end user, and Soft Solutions, Inc. By opening this package, you are agreeing to be bound to the terms of this agreement. If you do not agree to the terms of this agreement, promptly return the unopened disk package and the accompanying items (including written materials and binders or other containers) to the place you obtained them for a full refund.

Soft Solutions, Inc. Software License

1. **GRANT OF LICENSE.** Soft Solutions grants to you the right to use one development copy of the enclosed Soft Solutions software program (the "SOFTWARE") on a single terminal connected to a single computer (i.e., with a single CPU), or on a LICENSED COMPUTER NETWORK. Each concurrent user of the SOFTWARE must have exclusive access to a Soft Solutions SOFTWARE manual during his/her use. Soft Solutions, Inc. as Licensor, grants to you, the LICENSEE, a non-exclusive, nontransferable right to use this Software subject to the terms of the license as described in the following sections:

A. You may make backup copies of the SOFTWARE for your use provided they bear the Soft Solutions, Inc. copyright notice.

B. You may use this software in any number of custom or 4D-compiled non-commercial database applications created by the original licensee. An additional product deployment license may be required for each application running on 4D Server.

C. A separate licensing policy is required if you develop 4D-compiled commercial database applications which are sold to more than one entity. An entity refers to a person, company or association which pays to license your 4D application that utilizes SchedulePack. This licensing is based on an annual subscription fee set by Soft Solutions.

2. **COPYRIGHT.** The SOFTWARE is owned by Soft Solutions and is protected by United States copyright laws and international treaty provisions. Therefore, you must treat the SOFTWARE like any other copyrighted material (e.g., a book or musical recording) except that you may either (a) make one copy of the SOFTWARE solely for backup or archival purposes, or (b) transfer the SOFTWARE to a single hard disk provided you keep the original solely for backup or archival purposes. You may not copy the written materials accompanying the SOFTWARE.

3. **OTHER RESTRICTIONS.** You may not rent or lease the SOFTWARE, but you may transfer the SOFTWARE and accompanying written materials on a permanent basis provided you retain no copies and the recipient agrees to the terms of this Agreement. You may not reverse engineer, decompile, or decompile the SOFTWARE. If SOFTWARE is an update, any transfer must include the update and all prior versions.

**LIMITED WARRANTY.** Soft Solutions, Inc. warrants that (a) the SOFTWARE will perform substantially in accordance with the accompanying written materials for a period of ninety (90) days from the date of receipt. Any implied warranties on the SOFTWARE are limited to ninety (90) days and one (1) year, respectively. Some states do not allow limitations on duration of an implied warranty, so the above limitation may not apply to you.

**CUSTOMER REMEDIES.** Soft Solutions, Inc. entire liability and your exclusive remedy shall be, at Soft Solutions option, either (a) return of the price paid or (b) repair or replacement of the SOFTWARE that does not meet Soft Solutions Limited Warranty and which is returned to Soft Solutions with a copy of your receipt. This Limited Warranty is void if failure of the SOFTWARE has resulted from accident, abuse, or misapplication. Any replacement SOFTWARE will be warranted for the remainder of the original warranty period or thirty (30) days, whichever is longer.

**NO OTHER WARRANTIES.** Soft Solutions disclaims all other warranties, either express or implied, including but not limited to implied warranties of merchantability and fitness for a particular purpose, with respect to the SOFTWARE, the accompanying written materials, and any accompanying hardware. This limited warranty gives you specific legal rights. You may have others, which vary from state to state.

**NO LIABILITY FOR CONSEQUENTIAL DAMAGES.** In no event shall Soft Solutions or its suppliers be liable for any damages whatsoever (including, without limitation, damages for loss of business profits, business interruption, loss of business information, or other pecuniary loss) arising out of the use of or inability to use this Soft Solutions product, even if Soft Solutions has been advised of the possibility of such damages. Because some states do not allow the exclusion or limitation of liability for consequential or incidental damages, the above limitation may not apply to you.

**Governing Law.** This entire agreement shall be governed by the laws of the State of Georgia with the United States of America.

SchedulePack™ is a trademark of Soft Solutions, Inc. Apple is a registered trademark of Apple Computer, Inc. Windows is a registered trademark of Microsoft Corporation. 4th Dimension, 4D Server, 4D Compiler, 4D Engine and 4D are registered trademarks of 4D, Inc.



---

# Table of Contents

<b>Introduction .....</b>	<b>1</b>
Background: .....	1
Scheduling Problems Defined .....	1
SchedulePack Background and Design .....	1
One Form - Four Views .....	2
One Month View .....	2
Compatibility .....	3
Version 4 Feature Set .....	3
SchedulePack Programming Overview .....	4
SchedulePack Command Usage and Typical Order:.....	4
<b>4D Developer Methods.....</b>	<b>5</b>
Activation .....	5
Preferences .....	7
Banner Events.....	20
Contextual Menu .....	21
Configuration.....	22
Utilities .....	30
<b>Calendar Control Routines .....</b>	<b>41</b>
The SchedulePack Calendar Area .....	41
<b>Month View Commands .....</b>	<b>48</b>
The SchedulePack Month View .....	49
<b>Timeline Commands .....</b>	<b>53</b>
The SchedulePack Timeline Commands.....	53
<b>Chart Commands .....</b>	<b>56</b>
The SchedulePack Chart Commands .....	56
<b>Callback Reference .....</b>	<b>61</b>
Overview .....	61
Callback Variables.....	62
<b>Background Click Detection .....</b>	<b>65</b>
Action Codes .....	65

Incoming Action Codes .....	66
Outgoing Action Codes .....	75
<b>User Tips .....</b>	<b>75</b>
Creating Events/Objects .....	75
Deleting Events/Objects .....	76
Moving Events/Objects .....	76
Resizing Events/Objects .....	76
Coloring Events/Objects.....	76
Tabbing.....	76
Calendar Navigation .....	76
<b>Command Summary: .....</b>	<b>76</b>
Activation .....	76
Preferences .....	77
Banners .....	77
Configuration.....	77
Utilities .....	78
Calendar Control .....	78
Month View Control.....	79
Timeline View Control.....	79
Chart Control .....	79
CallBack Support Variables .....	80
Action Codes .....	80
Error Codes.....	81
<b>Programming Examples .....</b>	<b>85</b>
SchedulePack Definition and Setup.....	85
Typical Command Sequence.....	86
Application Definition and Setup .....	86
Declaration of Arrays for Related Resource Tables.....	87
Callback Variable Declaration .....	87
Action Variable Declaration.....	88
SchedulePack Area Views.....	90
Callback Examples .....	95

---

The Chart Commands .....	102
'SetupChart(ChartArea) .....	102
Chart_Resource_Setup .....	104
Color Assignment Options .....	106
The Calendar plug-in Area .....	107
The Calendar Area Script .....	108
SchedulePack Release History: .....	109
<b>Index .....</b>	<b>110</b>

## Table of Figures

Figure 1. SchedulePack One Month View display .....	2
Figure 2. Color Picker dialog .....	32
Figure 3. Import an iCal File.....	39
Figure 4. Add events .....	39
Figure 5. Timeline Commands.....	53
Figure 6. SchedulePack Chart Commands.....	56
Figure 7. The Activities table.....	85
Figure 8. Date View .....	90
Figure 9. Resource View .....	90
Figure 10. The OS X Color Wheel and 4D Color Grid .....	106
Figure 11. The Calendar plug-in Area .....	107



## Introduction

### Background:

SchedulePack is a 4th Dimension plug-in that provides compelling interactive graphical displays of calendar events and their associated resources. Resources are integral to events and include persons, places or things that are directed tied to the event. It is SchedulePack's internal management of these unique resources that distinguish it from general purpose calendars. Coupled with its careful attention to detail for solving a wide range of scheduling issues make SchedulePack an ideal centerpiece for 4D scheduling applications.

### Scheduling Problems Defined

Companies choosing to program a software scheduling solution with pure 4D code remain handcuffed with an underlying complexity. Experienced developers quickly realize how hard it is to simplify complex tasks. Consequently, many automated 4D-only scheduling solutions are burdensome to program and more burdensome to maintain. The result is often an inadequate, incomplete, and user-challenged application.

Scheduling solutions are needed in schools, medical offices, businesses, sporting events and countless organizations who require resources, such as people, rooms, etc. to be reserved for certain time windows on a certain day. For example, common to educational institutions is the classroom whose resources often include professors, students, equipment, rooms, etc. In a medical office or hospital setting, doctors, patients, rooms and assistants are all resources that must be continuously monitored and managed to order to avoid double-bookings and ensure the business is operated effectively. At a sports complex, playing fields, officials, scorekeepers and participants must all be carefully managed to ensure resources are singularly and humanely allocated at any one time. Because scheduled events often have mutually exclusive resources, proper allocation of these resources is essential to a successful scheduling. Viable scheduling is ultimately a combination of thoughtful resource allocation, integrity. SchedulePack's tools make this practical, possible and affordable while providing a new level of sophistication without the associated programming headaches. Developers will appreciate how tough scheduling assignments are simplified with SchedulePack.

### SchedulePack Background and Design

SchedulePack Events are contained in an Activity table whose records represent the primary currency of the SchedulePack plug-in. This table has been expanded in v4 with additional fields to provide tighter compatibility with Apple's iCal application. Because SchedulePack can use 4D records or array-based solutions, developers have two important development options. From concept, SchedulePack was designed to take advantage of the relational database engine of 4th Dimension. It shows related Events data (resources) in an attractive and informative manner to the end user.

Record adding, updating and deleting can optionally be handled directly by SchedulePack without any programming. Alternatively, callbacks can be used in conjunction with a complete set of Action Codes or mnemonics created from User-initiated actions. Through these callbacks, SchedulePack provides complete customizing capability.

SchedulePack, by default, will work with a specific "default" table name, "Events", defined internally. The 4D developer can name their table(s) and fields accordingly and no SchedulePack command is required. Alternatively, the 4D Developer can explicitly specify the Events Table to SchedulePack along with the required fields they intend to display.

SchedulePack's appearance and configuration are fully customizable in the 4D Design environment with the 4D Advanced Properties dialog. The 4D Developer can perform their own queries to establish the specific data a User will see within the SchedulePack area. The Developer can establish default background and text colors, row heights, column widths, fonts, font styles and font sizes of both the resource labels as well as the corresponding text areas. An interactive and customizable small Calendar external area is also provided within SchedulePack for simplifying the End-User date selection and manipulation. Developers can also show up to 3 SICN (small icons) per calendar day while defining the SICN's available to the application.

### One Form - Four Views

The SchedulePack v4 plug-in area has expanded to include the One Month, the Time-line, the Date and the Resource views on a single 4D form. This integration simplifies life for your end-users providing them an entire complement of scheduling tools at their fingertips. This integration also dramatically cuts down your development time while increasing your solution's value.

October 2009						
Sunday	Monday	Tuesday	Wednesday	Thursday	Friday	Saturday
27	28	29	30	1	2	3
				2 Day Therapy session		Eng201 at 9 AM on 9/13/09
4	5	6	7	8	9	10
• Hey There	• Meeting in room 2	• Phoenix Seminar (2 days) he • Hi There!	4 Day layover • New Red October Event	• Therapist 5 • This is event #1	• Hello, Hello	• New record • Here we go • Appointment • Monday the
11	12	13	14	15	16	17
• Therapist # • Morning ses • 1 Hour end • Open house • 45 minutes	• Weeklong forest retreat • Read 120 C • Here we gro • 1 Hour & 5C • Morning ses	• Duplicate Event	• Why n • 9-10 C • New Co • Read1 • 30 min	• 9-10 Color • True Blue • I Like this • 60 minutes	• Facial wrap should be wrapped a few times.	• Room 3; 2-3PM
18	19	20	21	22	23	24
It's a jungle out here • Relax in the pool... Why? • Morning Session Appoint	• Super Long Weekend event • 1 Hour 15 minutes	• Hi There! • Toby came! • Hey there!	• Long weekend session • Lobotomy is next on our l		• Geography • Hi There! In • summary n • Training ses • nothing	• Annual Dallas Conference • Back to the Future • Testing and Evaluation!
25	26	27	28	29	30	31
• How did it happen? Where • Where is it? How did it ha	• 2 day intensive • Dunder Mifflin Meeting	• Duplicate Event • test 1		• 9-10 Color ID=200 • Room 5	• Big Blue Ev • Hydrotherap • Therapist #	• Another mo • Hi There! • 2 Hours for • hello there

Figure 1. SchedulePack One Month View display

### One Month View

The SchedulePack One Month View provides an attractive drag and drop interface and includes a robust set of interactive Tools. Built on the standard monthly calendar look and feel, end-users can easily create, edit, delete and control-click activities to effectively manage their schedules. Other SchedulePack views include a resource, dates, time-line and chart view. Together, this collection of views elegantly solve a variety of scheduling problems in a productive user-friendly environment. Under v4, the new event-based contextual menus provide custom callback integration. This feature enables developers to create limitless options for the most demanding end-user requirements.

## **Compatibility**

SchedulePack 4.0 requires 4th Dimension 2004 or higher and includes full compatibility with 4th Dimension v11. SchedulePack is also compatible with all corresponding version levels of 4D Server. SchedulePack requires the Macintosh Operating System X or above and Windows 2000 or greater.

## **Version 4 Feature Set**

The traditional SchedulePack resource view now provides developers with integrated 4D Help tips, contextual menu support, background and foreground (text) color support, and compatibility with the Macintosh gradient quartz system and underlying OS Color wheel. New style sheets enable developers with simplified and centralized display formatting. Added support for 4D's Advanced Properties now allow developers to establish views with minimal coding. Also, under v4, pictures are supported for each day as well as for each event. Environmentally friendly, the added-drag and drop support is extended under v4 to support any 4D objects, including 3<sup>rd</sup> party plug-in objects. The above-mentioned capabilities are available for all views except the chart commands. This includes the SchedulePack resource and dates view as well as the new monthly and time-line views.

The v4 SchedulePack monthly view mirrors the functionality of Apple's iCal application, yet eliminates the need for duplicate calendars and events by harnessing the breadth and depth of 4D's underlying relational database. The SchedulePack monthly view is attractive, nimble, flexible and customizable. Underneath the hood, each days events are intelligently organized for one to many events. Governed by a linear programming-based events viewing model, the new monthly area includes support for multi-day banners, and compatibility with dragged and dropped objects from other 4D areas, including full integration with other 3<sup>rd</sup> party plug-ins, such as Arealist Pro.

While the array-based charting tool provides a read-only array based solution for a single days events, the SchedulePack time-line tool advances this concept into an broader, interactive tool for creating multi-day time-line solutions, including customizable display, icon support and callback support. With integrated Help tips and contextual menu support, businesses can simplify the complex. Here, developers can display one time grid including colorful and compelling time lines for each resource, while elegantly handling overlapping events.

Customized print capability gives the 4D Developer an excellent end user tool for printing one month calendars, multi-day time-line views, selected date range calendars and resource specific calendars in color.

Developers needing to manage multi-event/multi-task projects, including contingency-based events (incorporating Gant and similar chart tools) are encouraged to review Elegant, another Soft Solutions project scheduling plug-in built specifically to solve these type of problems.

Beginning in version 4, pictures are supported for each event as well. Multi-day banner events, with picture support is supported under v4. Through this capability, End Users can visually manage their time and ensure event conflicts are avoided. Full callback support allows the 4D Developer to customize their application specific to the scheduling requirements.

SchedulePack can be displayed as an independent, resizable external window to take full advantage of larger monitors. Version 4's Set View command enables 4D Developers to programmatically change calendar views from a daily, weekly, monthly or resource view with the same underlying external area. This eliminates countless man-hours of coding versus coding separate calendar areas. Version 4 provides 24 Bit Color Support, including OS X quartz gradient fills. V4 also provides enhanced drag and drop support to include non-SchedulePack areas.

## SchedulePack Programming Overview

The SchedulePack version 4 plug-in provides developers with a total of three plug-in areas and 6 calendar views. Developers can solve a variety of difficult and practical scheduling problems with simple date selection and display, a Scheduling area, rich with callback capabilities providing in-depth developer control and an improved interactive time-line area, providing customizable display and print capabilities. The charting tool, introduced in version 3, is ideal for providing end-users with a valuable timeline display.

There are approximately two dozen schedule related commands which provide for a detailed level of control over the SchedulePack area. However, only a subset of these commands are required to provide almost complete scheduling functionality. If you plan to use the Callback procedure to fully customize your scheduling application, there are a number of callback support neumerics (variables) readily available to you. **Before accessing the variables or using any SchedulePack commands, it is recommended you first declare the SchedulePack variables (as in 4D compiler declarations).** Failure to do so could cause unpredictable values and possibly crash 4D, especially when working in interpreted (uncompiled) mode. See the “Callback Support” section.

The most common and essential SchedulePack commands are:

- SXEventsTable
- SXCallbackProc
- SXBanners
- SXShowResources
- SXShowDates
- SXLabelsTable
- SXColumnsTable

There are nearly a dozen chart related commands which provide for a detailed level of control over charts and timelines.

## SchedulePack Command Usage and Typical Order:

The **SXEventsTable** function is first called to define the “events” table to the SchedulePack area. This is the table which contains the start and end dates, start and end times as well as the other details regarding the event. The **SXCallbackProc** must be called prior to displaying the SchedulePack plug-in area. This SchedulePack command defines the 4D method which is executed when the user performs keyboard actions on a SchedulePack plug-in area. Alternatively, if the developer does not require “callback” capability, they can pass a null string to specify that no callback method is enabled. The **SXBanners** command is called to specify whether or not a banner row is to be shown, and, if so, the physical row height. The **SXLabelsTable** is called prior to the **SXShowResources** or **SXShowDates** commands in order to enable the Resource labeling feature and identify the resource table, the ID field and the label field. The **SXColumnsTable** function is called prior to calling the **SXShowResources** command to identify the table and fields for the resource group that comprises the desired column definitions for the grid. The **SXShowDates** command is called to configure a SchedulePack area to display events for a specific range of dates, with a column on the grid for each date. The **SXShowResources** command is called to display events on a single date for a specific group of resources, with a column on the grid for each resource. Developers using the Timeline view can additionally use the **SXTimeLineOpt**, the **SXTimeLineFont** and the **SXTimeLineSetColor** commands to define the area.

**The SchedulePack commands shown above should initially be called where the first parameter is set to “0” (that’s zero).**

## 4D Developer Methods

### Activation

---

#### SXRegister

##### Description:

This function allows the 4D developer to register the SchedulePack program. SXRegister can be called in the start-up procedure of any 4D application using SchedulePack. A valid registration will return the current version of SchedulePack and fully enable the plug-in area. An invalid, empty or missing registration from the requesting machine's platform will return the string "INVALID REGISTRATION" or "INVALID DEPLOYMENT" and disable the plug-in area after 45 minutes. A valid registration returns the current SchedulePack version number (as found in the "Get info" window) of SchedulePack.

##### Syntax:

**SXRegister**(LicenseeName(S);LicenseCode Macintosh(S);LicenseCode Windows(S);  
Deployment License(S))->String(S)

##### Parameters:

*Licensee Name* ..... A string expression providing the name of the SchedulePack licensee.

*LicenseCode Macintosh* .... An string expression providing the matching Macintosh developer license code.

*LicenseCode Windows* ..... An string expression providing the matching Windows developer license code.

*Deployment License* ..... An string expression providing the 4D Client Server deployment license code. For non-OEM licensing (based on per Server deployment), Soft Solutions requires you first provide them with the 4D Server number where you intend to deploy SchedulePack. This server number will become the basis for generating the deployment license code. For OEM deployment (aka vertical market application or "blanket" deployment), the deployment license is created from the Licensee Name in lieu of the 4D Server Serial number. OEM licensing is annual, whereafter each year a purely informational message will appear after the OEM licensing period expires within the source code. Once a new OEM deployment license is issued, the informational messages will not appear until the OEM period expires. This message in no way effects the usage of the product.

##### Example:

**\$Version:=SXRegister**("Name","Mac Code","Windows Code","Deployment License Code") 'Registrations are issued by SSI.

---

#### SXShowDates

##### Description:

Use this routine to configure a SchedulePack area to display events for a specific range of dates. With

SXShowDates, each grid column contains the events for a single day. The ResourceIDs parameter (see 4th) should be used to specify (or constrain) the related resource records shown. For example, assume the ResourceIDs comprise arrays representing 4 of 8 total records in a related table. The SXShowDates retrieves and displays only those event records which are related to the records whose Unique ID's are contained within the ResourceID's array. In conjunction with the SXColumnsTable command, SchedulePack searches for both the Event label table (the table which contains the label for each cell) and the column Tag Table.

**Syntax:**

**SXShowDates**(xArea(L);StartDate(D);EndDate(D);ResourceIDs(Array L))->L

**Parameters:**

- xArea*..... Pass the value of a variable reference to a SchedulePack external-area object on a currently-active layout.
- StartDate*..... A date expression which provides the date for the first column in the grid.
- EndDate*..... A date expression which provides the date for the last column in the grid.
- ResourceIDs*..... A long-integer array. If you have already called the **SXColumnsTable** command to define the column-resource table, and wish to limit the display of events on this grid to events linked to one or more of the resources in this table, pass the ID number(s) of the desired resource record(s) as the values of the elements of this array.
- If you wish to display all events regardless of their resource assignments, pass an empty array.

**Error Codes:**

- 0 - Successful call made
- 15700 - Invalid SchedulePack Area
- 15300 - Invalid Array Type - Not Long Integer
- 15400 - Invalid Date

**Example:**

**\$x:=SXShowDates**(xSch;Current Date;Current Date+6;aDoctorIDs)

---

## SXShowResources

**Description:**

Use this function to configure a SchedulePack area to display events on a single date for a specific group of resources, with a column on the grid for each resource. Prior to calling this routine, you must use the **SXColumnsTable** command to identify the resource table and fields which identify the resources to use for the columns. To show all events associated with any (or no) resource, you must pass an empty array.

**Syntax:**

**SXShowResources**(xArea(L);Date(D);ResourceIDs(Array L))->L

**Parameters:**

- xArea*..... Pass the value of a variable reference to a SchedulePack external-area object on a currently-active layout.
- Date*..... A date expression which provides the single date for the grid.
- ResourceIDs*..... A long-integer array which contains an element for each desired column in the grid. The value of each element should be the record ID of the desired resource.

**Error Codes :**

- 0 - Successful call made
- 15700 - Invalid SchedulePack Area
- 15300 - Invalid Array Type - Not Long Integer

**Example:**

**\$x:=SXShowResources(xSch;Current Date;aDoctorIDs)**

**Preferences**

---

**SXRowHdrFont**

**Description:**

Allows the developer to assign the font, size and style of text used for the row headers (the time-slot labels).

**Syntax:**

**SXRowHdrFont(xArea(L);FontName(S);FontSize(I);FontStyle(I),RowHeight(I))**

**Parameters:**

- xArea*..... A Long Integer containing either the value of a variable reference to a SchedulePack external-area object on a currently-active layout, or zero. If you pass a reference value, the command will apply only to that object. If you pass zero, the command will apply to the defaults for all future (as yet undisplayed) objects.
- FontName*..... A string expression providing the name of the desired font (which should be resident on the user's system).
- FontSize* ..... An integer expression providing the desired font size.
- FontStyle* ..... An integer expression providing the desired font styles (encoded using the same values as 4D's FONT STYLE command).  
0 = Plain, 1 = Bold, 2 = Italic, 4 = Underline, 8 = Outline,  
16 = Shadow, 32 = Condensed, 64 = Extended
- RowHeight* ..... An integer expression specifying the height of rows in pixels.  
0 = automatically adjust row heights,  
0 >= fixed amount in pixels,  
-1 = retains current setting.
- FontColor*..... A long integer expression representing the 24 bit color (*new in v4*).

**Example:**

**SXRowHdrFont**(xSch;"Geneva";14;1,30) '14 point Geneva Bold

The following special values can be passed to the SXRowHdrFont command so that it will retain the current setting:

FontName = ""  
FontSize = -1  
FontStyle = -1

**Example:**

**SXRowHdrFont**(xArea;"Palatino";-1;2) `make Font Palatino,don't change size, make style Italic

---

## SXColHdrFont

**Description:**

Allows the developer to assign the font, size and style of the text used for the column headers (the date/resource labels).

**Syntax:**

**SXColHdrFont**(xArea(L);FontName(S);FontSize(I);FontStyle(I))

**Parameters:**

- xArea*..... Pass either the value of a variable reference to a SchedulePack external-area object on a currently-active layout, or zero. If you pass a reference value, the command will apply only to that object. If you pass zero, the command will apply to the defaults for all future (as yet undisplayed) objects.
- FontName*..... A string expression providing the name of the desired font (which should be resident on the user's system).
- FontSize* ..... An integer expression providing the desired font size.
- FontStyle* .....An integer expression providing the desired font styles (encoded using the same values as 4D's FONT STYLE command).  
0 = Plain, 1 = Bold, 2 = Italic, 4 = Underline, 8 = Outline,  
16 = Shadow, 32 = Condensed, 64 = Extended
- FontColor*..... A long integer expression representing the 24 bit color (*new in v4*).

**Example:**

**SXColHdrFont**(xSch;"Geneva";14;2) '14 point Geneva Italicized

The following special values can be passed to the SXColHdrFont command so that it will retain the current setting:

FontName = ""  
FontSize = -1  
FontStyle = -1

**Example:**

`SXColHdrFont(xArea;"";12;4)` don't change Font,make size 12, make style Underline`

**SXEventFont****Description:**

Allows the developer to assign the font, size and style of the text used for the editable event summaries.

**Syntax:**

`SXEventFont(xArea(L);FontName(S);FontSize(I);FontStyle(I);EventColor(L))`

**Parameters:**

- xArea*..... Pass either the value of a variable reference to a SchedulePack external-area object on a currently-active layout, or zero. If you pass a reference value, the command will apply only to that object. If you pass zero, the command will apply to the defaults for all future (as yet undisplayed) objects.
- FontName*..... A string expression providing the name of the desired font (which should be resident on the user's system).
- FontSize* ..... An integer expression providing the desired font size.
- FontStyle* ..... An integer expression providing the desired font styles (encoded using the same values as 4D's FONT STYLE command).  
 0 = Plain, 1 = Bold, 2 = Italic, 4 = Underline, 8 = Outline,  
 16 = Shadow, 32 = Condensed, 64 = Extended
- EventColor* ..... A long integer expression containing the 24 bit color desired.  
 A value of -1 will retain previous color setting (*new in v4*).

**Example:**

`SXEventFont(xSch;"Monaco";12;0;-1) '12 point Monaco Plain, retain color`

The following special values can be passed to the SXEventFont command so that it will retain the current setting:

```
FontName = ""
FontSize = -1
FontStyle = -1
EventColor = -1
```

**Example:**

`SXEventFont(xArea;"Monaco";9;-1;-1)` make Font Monaco, make size 9, don't change style or color`

## SXEvtLabelFont

**Description:**

Allows the developer to assign the font, size, style and color of the text used for the static event-resource labels.

**Syntax:**

**SXEvtLabelFont**(xArea(L);FontName(S);FontSize(I);FontStyle(I); FontColor(L))

**Parameters:**

- xArea*..... Pass either the value of a variable reference to a SchedulePack external-area object on a currently-active layout, or zero. If you pass a reference value, the command will apply only to that object. If you pass zero, the command will apply to the defaults for all future (as yet undisplayed) objects.
- FontName*..... A string expression providing the name of the desired font (which should be resident on the user's system).
- FontSize* ..... An integer expression providing the desired font size.
- FontStyle* ..... An integer expression providing the desired font styles (encoded using the same values as 4D's FONT STYLE command).  
0 = Plain, 1 = Bold, 2 = Italic, 4 = Underline, 8 = Outline,  
16 = Shadow, 32 = Condensed, 64 = Extended
- FontColor*..... A long integer expression containing the 24 bit color desired.  
A value of -1 will retain previous color setting (*new in v4*).

**Example:**

**SXEvtLabelFont**(xSch;"Monaco";12;2;-1)

The following special values can be passed to the SXEvtLabelFont command so that it will retain the current setting:

FontName = ""  
FontSize = -1  
FontStyle = -1  
FontColor = -1

**Example:**

**SXEvtLabelFont**(xArea;"";12;4;-1) don't change font or color,make size 12, make style Underline

---

## SXMinColWidth

**Description:**

Allows the developer to define the desired minimum column width. The actual column width may be larger, but not smaller.

**Syntax:**

**SXMinColWidth**(xArea(L);ColWidth(I))

**Parameters:**

- xArea*..... Pass either the value of a variable reference to a SchedulePack external-area object on a currently-active layout, or zero. If you pass a reference value, the command will apply only to that object. If you pass zero, the command will apply to the defaults for all future (as yet undisplayed) objects.
- ColWidth*..... An integer expression providing the desired minimum column width in pixels (where 72 pixels equal one inch).

**Example:**

**SXMinColWidth**(xSch;36) 'Search column to 36 pixels (1/2 inch) or more

---

**SXDateFormat**

**Description:**

Allows the developer to specify the date formatting option to use when labeling the columns in a date-column grid.

**Syntax:**

**SXDateFormat**(xArea(L);FormatCode(I))

**Parameters:**

- xArea*..... Pass either the value of a variable reference to a SchedulePack external-area object on a currently-active layout, or zero. If you pass a reference value, the command will apply only to that object. If you pass zero, the command will apply to the defaults for all future (as yet undisplayed) objects.
- FormatCode*..... An integer expression providing the desired formatting option. The following values are defined:
  - 0 - Use the default format, with the date's day of the month left-justified in the col-umn-header box, and the weekday-name right-justified.
  - All other values correspond to the format-codes used by 4D's String function, with the resulting string centered in the column-header box:
    - 1 - Short (8/16/98)
    - 2 - Abbreviated (Sun, Aug 16, 1998)
    - 3 - Long (Sunday, August 16, 1998)
    - 4 - mm/dd/yyyy (08/16/92 or 08/16/1898)
    - 5 - Month Date, Year (August 16, 1998)
    - 6 - Abbr. Month Date, Year (Aug 16, 1998)
    - 7 - Abbreviated without weekday (Aug 16, 1998)

**Example:**

`SXDateFormat(xSch;1)`

To see an example, please refer to the “SchedulePack Area ” on page 90.

---

**SXTimeFormat**

**Description:**

Allows the developer to specify the time formatting option to use when labeling the rows in a SchedulePack grid.

**Syntax:**

`SXTimeFormat(xArea(L);TimeCycle(I);SuppressSuffix(I))`

**Parameters:**

*xArea*..... Pass either the value of a variable reference to a SchedulePack external-area object on a currently-active layout, or zero. If you pass a reference value, the command will apply only to that object. If you pass zero, the command will apply to the defaults for all future (as yet undisplayed) objects.

*TimeCycle* ..... An integer expression providing the desired time cycle or interval. The following values are defined:  
0 - Displays either Hour or Minutes (but not both)  
1 - Use format from Operating System Control Panel  
2 - Use 24-hour format midnight = 00:00  
3 - Use A.M./P.M. format midnight = 00:00  
4 - Use A.M./P.M. format midnight = 12:00

*SuppressSuffix* ..... An integer expression that displays or hides the AM/PM suffix. The following values are defined:  
0 - Don't suppress suffix.  
1 - Suppress suffix;

NOTE:  
TimeCycle = 0 overrides this option.

**Example:**

`SXTimeFormat(xSch;2;1)` 'Display 30 minute intervals and suppress the AM/PM suffix.

---

**SXSetColor**

**Description:**

Allows the developer to set color preferences for up to 12 separate SchedulePack areas.

**Syntax:**

`SXSetColor(xArea(L);Part(I);Color(L))`

**Parameters:**

- xArea*..... Pass either the value of a variable reference to a SchedulePack external-area object on a currently-active layout, or zero. If you pass a reference value, the command will apply only to that object. If you pass zero, the command will apply to the defaults for all future (as yet undisplayed) objects.
  
- Part* ..... An integer expression representing a SchedulePack area (there are 12 options). The following values are defined:
  - 1 – Column Headers background,
  - 2 – Row Headers Background for non-business,
  - 3 - Row Headers Background for business,
  - 4 – Banner Header Background,
  - 5 – Banner Background,
  - 6 – Grid Background, - business
  - 7 - Grid Background, - non business,
  - 8 - Hour grid lines,
  - 9 – Half-hour grid lines,
  - 10 – Quarter-hour grid lines,
  - 11 – Minor hour grid lines,
  - 12 – Column grid lines
  
- 24bitColor*..... A long integer expression containing the 24 bit color desired. A value of -1 will retain previous color setting (*new in v4*).

**Example:**

```
SXSetColor(xSch;5;-1)
```

---

**SXAreaOpt**

**Description:**

Allows the developer to enable or disable horizontal and vertical scrolling as well as specifying the type of tracking preference.

**Syntax:**

```
SXAreaOpt(xArea(L);SetHorizScroll(I);SetVertScroll(I);RealEventTrack(I))
```

**Parameters:**

- xArea*..... Pass either the value of a variable reference to a SchedulePack external-area object on a currently-active layout, or zero. If you pass a reference value, the command will apply only to that object. If you pass zero, the command will apply to the defaults for all future (as yet undisplayed) objects.
  
- SetHorizScroll*..... An integer expression hiding or displaying the horizontal scroll bars. The following values are defined:
  - 0 - Remove the horizontal scroll bar from the SchedulePack display area.
  - 1 - Show the vertical scroll bar in the SchedulePack display area.

-1 - Retains current setting.

*SetVertScroll*..... An integer expression hiding or displaying the vertical scroll bars.  
The following values are defined:  
0 - Remove the vertical scroll bar from the SchedulePack display area.  
1 - Show the vertical scroll bar in the SchedulePack display area.  
-1 - Retains current setting.

*RealEventTrack*..... An integer expression enabling or disabling the real time event tracking.  
Set this parameter to 0 to have the Event object physically move with the Users mouse dragging action. Set this parameter to 1 to have dotted lines physically move with the Users mouse dragging action. The following values are defined:  
0 - Disable real time event tracking.  
1 - Enable real time event tracking.  
-1 - Retains current setting.

**Example:**

**SXAreaOpt(xSch;1;1;1)**

---

## SXEventOpt

**Description:**

Allows the developer to enable or disable Event and Banner creation as well as using the color pop-up for Events and Banners.

**Syntax:**

**SXEventOpt**(xArea(L);CanMakeBanner(I);CanColorBanner(I);CanMakeEvent(I);CanColor-Event(I);OverlapIndentation(I),EventRoundAmount(I);UseGradientFills(I);UseToolTips(I))

**Parameters:**

*xArea*..... Pass either the value of a variable reference to a SchedulePack external-area object on a currently-active layout, or zero. If you pass a reference value, the command will apply only to that object. If you pass zero, the command will apply to the defaults for all future (as yet undisplayed) objects.

*CanMakeBanner* ..... An integer expression enabling or disabling Banner creation capability:  
0 - Disable Banner creation capability within SchedulePack area.  
1 - Enable Banner creation capability within SchedulePack area.  
-1 - Retains current setting.

*CanColorBanner*..... An integer expression allowing or disallowing color pop-up for a Banner:  
0 - Disable Banner color pop-up menu within SchedulePack area.  
1 - Enable Banner color pop-up menu within SchedulePack area.  
-1 - Retains current setting.

- CanMakeEvent*..... An integer expression allowing or disallowing color pop-up for an Event. The following values are defined:  
0 - Disable event creation capability within SchedulePack area.  
1 - Enable event creation capability within SchedulePack area.  
-1 - Retains current setting.
- CanColorEvent* ..... An integer expression to enable or disable the color-popup for event objects. The following values are defined:  
0 - Disable event color pop-up menu within SchedulePack area.  
1 - Enable event color pop-up menu within SchedulePack area.  
-1 - Retains current setting.
- Overlap Indentation*..... An integer expression specifying how many pixels an overlapping event object will indent to preserve visibility. The default value is 12, for 12 pixels indentation. The following values are defined:  
>0 -The number of pixels an overlapping event object will indent.  
-1 - Retains current setting.
- EventRoundAmount*.....An integer expression specifying the rounding amount of Events' corners.  
0 - No rounding  
>0 - Where 1 is minimum rounding, and 100 is maximum rounding.  
-1 - Retains current setting.
- UseGradientFill*..... An integer expression where 0=no and 1=yes (new in v4).  
-1 - Retains current setting.
- UseToolTips*..... An integer expression where 0=no and 1=yes (new in v4).  
-1 - Retains current setting.

**Example:**

**SXEventOpt**(xSch;1;1;1;12,25;1;1)

---

**SXDragCreateOpt**

**Description:**

Allows the developer to enable or disable Event and Banner creation through point, click, drag and release methodology.

**Syntax:**

**SXDragCreateOpt**(xArea(L);CanMakeWideBnr(I);CanMakeWideEvt(I);CanMakeTallEvt(I))

**Parameters:**

*xArea*..... Pass either the value of a variable reference to a SchedulePack external-area object on a currently-active layout, or zero. If you pass a

reference value, the command will apply only to that object. If you pass zero, the command will apply to the defaults for all future (as yet undisplayed) objects.

- CanMakeWideBnr*..... An integer expression enabling or disabling multi-day (wide) banner creation capability:
  - 0 - Disable multi-day banner creation capability within SchedulePack area.
  - 1 - Enable multi-day banner creation capability within SchedulePack area.
  - 1 - Retains current setting.
  
- CanMakeWideEvt* ..... An integer expression allowing or disallowing multi-day Event creation:
  - 0 - Disable multi-day event creation capability within SchedulePack area.
  - 1 - Enable multi-day event creation capability within SchedulePack area.
  - 1 - Retains current setting.
  
- CanMakeTallEvent* ..... An integer expression to enable or disable the creation of multi-cell objects or “Tall” event objects. The following values are defined:
  - 0 - Disable “Tall” event creation capability within SchedulePack area.
  - 1 - Enable “Tall” event creation capability within SchedulePack area.
  - 1 - Retains current setting.

**Example:**

**SXDragCreateOpt(xSch;1;1;0)**

---

## SXDragMoveOpt

**Description:**

Allows the developer to enable or disable Event and Banner moving options.

**Syntax:**

**SXDragMoveOpts(xArea(L);CanDragBanner(I);CanDragHorizEvt(I);CanDragVertEvt(I))**

**Parameters:**

- xArea*..... Pass either the value of a variable reference to a SchedulePack external-area object on a currently-active layout, or zero. If you pass a reference value, the command will apply only to that object. If you pass zero, the command will apply to the defaults for all future (as yet undisplayed) objects.
  
- CanDragBanner*..... An integer expression enabling or disabling the moving of a multi-day (wide) banner:
  - 0 - Disable multi-day banner moving capability within SchedulePack area.
  - 1 - Enable multi-day banner moving capability within SchedulePack area.
  - 1 - Retains current setting.
  
- CanDragHorizEvt* ..... An integer expression allowing or disallowing moving for a wide Event:
  - 0 - Disable multi-day event moving capability within SchedulePack area.
  - 1 - Enable multi-day event moving capability within SchedulePack area.

-1 - Retains current setting.

*CanDragVertEvt*..... An integer expression to enable or disable the moving of “Tall” event objects. The following values are defined:

0 - Disable “Tall” event moving capability within SchedulePack area.

1 - Enable “Tall” event moving capability within SchedulePack area.

-1 - Retains current setting.

**Example:**

**SXDragMoveOpt**(xSch;1;1;0)

---

**SXDragResizeOpt**

**Description:**

Allows the developer to enable or disable Event and Banner resizing options.

**Syntax:**

**SXDragResizeOpt**(xArea(L);CanResizeWideBnr(I);CanResizeWideEvt(I);CanResizeTallEvt(I))

**Parameters:**

*xArea*..... Pass either the value of a variable reference to a SchedulePack external-area object on a currently-active layout, or zero. If you pass a reference value, the command will apply only to that object. If you pass zero, the command will apply to the defaults for all future (as yet undisplayed) objects.

*CanResizeWideBnr* ..... An integer expression enabling or disabling the resizing of a multi-day (wide) banner:

0 - Disable multi-day banner resizing capability within SchedulePack area.

1 - Enable multi-day banner resizing capability within SchedulePack area.

-1 - Retains current setting.

*CanResizeWideEvt*..... An integer expression allowing or disallowing resizing for a wide Event:

0 - Disable multi-day event resizing capability within SchedulePack area.

1 - Enable multi-day event resizing capability within SchedulePack area.

-1 - Retains current setting.

*CanResizeTallEvent*..... An integer expression to enable or disable the resizing of multi-cell or “Tall” event objects. The following values are defined:

0 - Disable “Tall” event resizing capability within SchedulePack area.

1 - Enable “Tall” event resizing capability within SchedulePack area.

-1 - Retains current setting.

**Example:**

**SXDragResizeOpt**(xSch;1;1;0)

## SXCreateStyleSheet

**Description:**

Allows the developer to create a stylesheet with Font name, Font size, Font style and Font color. The stylesheet can then be used in a call to 'SXApplyStyleSheet'.

**Syntax:**

**SXCreateStyleSheet** (xArea(L);ID(L);FontName(S);FontSize(I);FontStyle(I); FontColor(L)) ->L

**Parameters:**

- xArea*..... Pass the value of a variable reference to a SchedulePack external-area object on a currently-active layout.
- ID* ..... A long integer expression that is to be used as the ID for the stylesheet.
- FontName*..... A string expression providing the name of the desired font (which should be resident on the user's system).
- FontSize* ..... An integer expression providing the desired font size.
- FontStyle* ..... An integer expression providing the desired font styles (encoded using the same values as 4D's FONT STYLE command).  
0 = Plain, 1 = Bold, 2 = Italic, 4 = Underline, 8 = Outline,  
16 = Shadow, 32 = Condensed, 64 = Extended
- FontColor*..... A long integer expression containing the 24 bit color desired.

**Error Codes:**

- 0 - Successful call made
- 15700 - Invalid SchedulePack Area
- 15410 - Invalid Font name
- 15411 - Invalid Font size
- 15412 - Invalid Font style
- 15413 - Invalid Font color

**Example:**

```
$StyleSheetID:=123456  
$x:=SXCreateStyleSheet (xSch;$StyleSheetID;"Monaco";12;2;$Color)
```

---

## SXApplyStyleSheet

**Description:**

Applies a stylesheet's properties to a specific SchedulePack area part.

**Syntax:**

**SXApplyStyleSheet** (xArea(L);ID(L);Part(I)) ->L

**Parameters:**

- xArea*..... Pass the value of a variable reference to a SchedulePack external-area object on a currently-active layout.

*ID*..... A long integer expression that is the ID of the stylesheet.

*Part*..... An integer expression indicating the day of week.

- 1 - Month & Year
- 2 - Weekday Names
- 3 - Day Numbers
- 4 - non-month day numbers
- 5 - Events
- 6 - Event Labels
- 7 - Row Headers
- 8 - Column Headers

**Error Codes:**

- 0 - Successful call made
- 15700 - Invalid SchedulePack Area
- 15420 - Invalid stylesheet ID

**Example:**

`$x:=SXApplyStyleSheet(xSch;$StyleSheetID;3)`

---

## SXDeleteStyleSheet

**Description:**

Deletes a stylesheet from memory.

**Syntax:**

`SXDeleteStyleSheet (xArea(L);ID(L)) ->L`

**Parameters:**

*xArea*..... Pass the value of a variable reference to a SchedulePack external-area object on a currently-active layout.

*ID*..... A long integer expression that is the ID of the stylesheet.

**Error Codes:**

- 0 - Successful call made
- 15700 - Invalid SchedulePack Area
- 15420 - Invalid stylesheet ID

**Example:**

`$x:=SXDeleteStyleSheet(xSch;$StyleSheetID)`

## Banner Events

Banner events are special events that are distinguished from appointment events by the value of the “Event Type” field in the Events table. When this field is equal to 1 for a given event record, that event will be drawn in a special row at the top of the grid.

This row will be above the time-slot row for the first time interval on the grid, and will be labeled with an arbitrary symbol (gray diamond) in the left-hand border. The values of the start and end times stored for the event will be ignored and the event box will not be allowed to overlap the row boundary for this special row. Banner events can overlap column boundaries when displayed on a date-column grid; in other words, the end date may be greater than the start date.

When the user tries to move or resize a banner event, they will be restricted to the boundaries of the banner-event row. They will be able to move the event box left or right, in order to reschedule the start date, and they will be able to “stretch” the right border left or right in order to change the end date.

At no time will user actions or any SchedulePack function have any effect on the start and end times for a banner event. When a user creates a banner event in the grid, the start and end times will be initialized to zero. The application can make changes to the values in these fields, or allow the user to edit them via the callback procedure, and such changes will be respected (and ignored) by SchedulePack.

---

## SXBannerRow

### Description:

Allows the developer to define the desired height of the special row at the top of the grid where banner events are displayed and edited. The default height of this row is equal to twice the derived height of the regular time-interval rows. If you do not wish to display or allow users to create banner events, specify a height value of zero.

### Syntax:

**SXBannerRow**(*xArea*(L);BannerRowHt(I))

### Parameters:

*xArea*..... Pass either the value of a variable reference to a SchedulePack external-area object on a currently-active layout, or zero. If you pass a reference value, the command will apply only to that object. If you pass zero, the command will apply to the defaults for all future (as yet undisplayed) objects.

*BannerRowHt*..... An integer expression providing the desired height of the banner-events row in pixels.

### Example:

**SXBannerRow**(*xSch*;0) `Pass 0 to prevent banners from being displayed

**SXBannerRow**(*xSch*;-1) `Pass -1 to allow SchedulePack to calculate default height

**SXBannerRow**(*xSch*;144) `Displays a 2 inch high (144 pixels) Banner row

To see an example, please refer to the “[SchedulePack Area](#)” on page 90.

## Contextual Menu

Banner A contextual menu will be displayed when holding down the control key and clicking on an event or a date/resource cell.

The contextual menu when clicking on an event will have these menu items:

- Cut
- Copy
- Paste
- Duplicate
- Delete

The contextual menu when clicking on a date/resource cell will have these menu items:

- New Event
- Paste Event

Custom menu items and sub-menus can be dynamically added to the contextual menu through the SchedulePack callback method. Before the menu is displayed the callback will receive an action value of 'SXkContextualBefore'. A two-dimensional text array named 'SX\_APopupMenuItems' can be populated. Column zero of each row will be added to the main menu. Each row can have sub-menu by using an array declaration on the row.

### Example:

```
ARRAY TEXT(SX_APopupMenuItems;4;0) ` declare the array to have 4 rows (menu items)
```

```
` assign the values for the items on the main menu
```

```
SX_APopupMenuItems{1}{0}:="Therapists"
```

```
SX_APopupMenuItems{2}{0}:="Rooms"
```

```
SX_APopupMenuItems{3}{0}:="Courses"
```

```
SX_APopupMenuItems{4}{0}:="Equipment"
```

```
` add a "Rooms" sub-menu
```

```
$roomsRow:=2
```

```
ARRAY TEXT(SX_APopupMenuItems{$roomsRow};5) ` declare the "Rooms" row to have 5 elements
```

```
SX_APopupMenuItems{$roomsRow}{1}:="Room 1"
```

```
SX_APopupMenuItems{$roomsRow}{2}:="Room 2"
```

```
SX_APopupMenuItems{$roomsRow}{3}:="Room 3"
```

```
SX_APopupMenuItems{$roomsRow}{4}:="Room 4"
```

```
SX_APopupMenuItems{$roomsRow}{5}:="Room 5"
```

After the contextual menu is displayed and the user has made a menu item choice the callback will

receive an action value of 'SXkContextualAfter'. Two variables, 'SX\_PopupDevRow' & 'SX\_PopupDevCol', will contain the row number and column number respectively of the user's choice. e.g. - if the user chose "Room 5" then SX\_PopupDevRow will be a value of 2, and SX\_PopupDevCol will be a value of 5.

## Configuration

---

### SXTimeIntervals

#### Description:

Allows the developer to assign the overall start and end times for the grid, the start and end times for the grid's "business hours", and the intervals per hour for the SchedulePack grid.

The time values provided for the Start, End, BHStart and BHEnd parameters are all expressed as a time or a long integer. With version 3, it is now possible to pass a long integer, representing the seconds since midnight. The calculation is as follows, for 10 AM, the 4D Developer could pass (10 hours x 60 minutes x 60 seconds) or 36,000. This represents the number of seconds since midnight.

Additionally, for an end time extending into the AM hours following midnight from a previous day, the developer would pass the number of seconds since midnight on the previous day. So, for example, for 2 AM, a developer would pass 26x60x60 or 93,600

The interval mode should be an integer which is evenly divisible into 60. If the developer passes a number which is not divisible into 60, SchedulePack will automatically round the number passed upwards into the next number which is evenly divisible into 60.

#### Syntax:

**SXTimeIntervals**(xArea(L);Start(H/L);End(H/L);BHStart(H/L);BHEnd(H/L);IntervalMode(I))->L

#### Parameters:

*xArea*..... Pass either the value of a variable reference to a SchedulePack external-area object on a currently-active layout, or zero. If you pass a reference value, the command will apply only to that object. If you pass zero, the command will apply to the defaults for all future (as yet undisplayed) objects.

*Start*..... A time or long integer expression providing the desired starting time for the grid.

*End*..... A time or long integer expression providing the desired ending time for the grid.

*BHStart* ..... A time or long integer expression providing the desired starting time for the "business hours" portion of the grid.

*BHEnd*..... A time or long integer expression providing the desired ending time for the "business hours" portion of the grid.

*IntervalMode*..... An integer expression specifying the desired interval mode (the duration for each row) for the grid.

Valid values for the interval mode are:

1 - 1 row per hour (1-hour cells).

2 - 2 rows per hour (30-min. cells).

- 4 - 4 rows per hour (15-min. cells).
- 5 - 5 rows per hour (12-min. cells).
- 6 - 6 rows per hour (10-min. cells).
- 10 - 10 rows per hour (6-min. cells).
- 12 - 12 rows per hour (5-min. cells).
- 15 - 15 rows per hour (4-min. cells).
- 20 - 20 rows per hour (3-min. cells).
- 30 - 30 rows per hour (2-min. cells).

**Error Codes:**

- 0 - Successful call made
- 15700 - Invalid SchedulePack Area
- 15401 - Invalid Time
- 15402 - Invalid Mode

**Example:**

```
$X:=SXTimeInterval(xSch;†09:00:00†;†17:00:00†;†09:00:00†;†17:00:00†;1)
```

---

**SXCallbackProc**

**Description:**

Allows the developer to assign the name of the 4D callback procedure, i.e. the global procedure to be called whenever any of a list of key events occur. The callback procedure must declare a boolean field for the \$0 parameter “C\_Boolean(\$0)”. The developer will set \$0 equal to True when it wants SchedulePack to automatically perform its update. If the developer desires to perform all updates independent of SchedulePack, \$0 should be set to False.

**Syntax:**

```
SXCallbackProc(xArea(L);CallbackProc(S))
```

**Parameters:**

- xArea*..... Pass either the value of a variable reference to a SchedulePack external-area object on a currently-active layout, or zero. If you pass a reference value, the command will apply only to that object. If you pass zero, the command will apply to the defaults for all future (as yet undisplayed) objects.
- CallbackProc* ..... A string expression providing the name of an existing 4D procedure. The 4D procedure specified must be written to require no parameters, and must return a boolean result.  
 Specifying a procedure that does not match these syntax requirements will result in a 4D runtime error or system crash.

**Example:**

```
SXCallbackProc(xSch;"SP_Callback")
```

To see an example, please refer to the “[Application Definition and Setup](#)” on page 86.

## SXEventsTable

### Description:

This function enables the 4D developer to specify the database table used to store the event descriptions created and displayed by a SchedulePack external object. In order to use the SchedulePack package, the application must provide it with a database table in which to store event descriptions.

The table must contain, at a minimum, the following fields:

Record ID - Long Integer, Indexed & Unique  
Summary - Alpha 80  
Start Date - Date, Indexed  
End Date - Date, Indexed  
Start Time - Time, Indexed  
End Time - Time, Indexed  
Color Index as Integer, or 24-bit Color as Longint  
Event Type - Integer

The field names used above are only defaults - your application can use any name you wish for the table itself and for these fields within it. The datatypes and attributes given for these fields, on the other hand, are required.

The fields can be in any order in the table definition, and the table can have any number of other fields, so long as none of them have the "Mandatory" attribute.

If the 4D database has a table named "Events", which has fields with the names and attributes shown above, and you wish to use this table for storing SchedulePack's event items, then this command is not needed - the package searches the database for such a table at start-up, and if found, will use it by default.

The best way to obtain the table number for a given table is to pass a pointer to the table as the parameter to 4D's Table function, as in:

```
$fileNo:=File(»[MyEventsFile]) 'v3 Syntax
```

```
$tableNo:=Table(->[MyEventsTable]) 'v6 Syntax
```

The best way to obtain the field number for a given field is to pass a pointer to the field as the parameter to 4D's Field function, as in: `$fieldNo:=Field(->[MyEventsTable]RecordID)`

These fields are the only ones required to use SchedulePack for most basic applications; some of SchedulePack's more advanced features require additional fields in the Events table and other tables in the database. Refer to the commands SXLabelsTable on page 26 and SXColumnsTable on page 27 for more information.

### Syntax:

```
SXEventsTable(xArea(L);EventsTableNo(I);RecIDFld(I);StartDtFld(I);EndDtFld(I);StartTmFld(I);  
EndTmFld(I);SummaryFld(I);ColorFld(I);EventType(I);TextColorFld(I);IconFld(I))->L
```

### Parameters:

*xArea*..... Pass either the value of a variable reference to a SchedulePack external-area object on a currently-active layout, or zero. If you pass a

reference value, the command will apply only to that object. If you pass zero, the command will apply to the defaults for all future (as yet undisplayed) objects.

- EventsTableNo*..... For the events table field an integer expression which provides the table number for the database table in which the specified SchedulePack object is to find and store its event descriptions.
- RecIDFld*..... An integer expression which provides the field number for the events table field (as specified by EventsTableNo) which will store the record ID's for each event.  
The record ID is a unique, arbitrary key which will be used to identify individual events by SchedulePack. The field's type must be **Long Integer**, and it should be indexed and unique.
- StartDtFld* ..... An integer expression which provides the field number for the events table field (as specified by EventsTableNo) which will store the starting dates for each event. The field's type must be **Date**, and it should be indexed.
- EndDtFld* ..... An Integer expression which provides the field number for the events table field (as specified by EventsTableNo) which will store the ending dates for each event. The field's type must be **Date**, and it should be indexed.
- StartTmFld*..... An integer expression which provides the field number for the events table field (as specified by EventsTableNo) which will store the starting times for each event. The field's type must be **Time**, and it should be indexed.
- EndTmFld* ..... An integer expression which provides the field number for the events table field (as specified by EventsTableNo) which will store the ending times for each event. The field's type must be **Time**, and it should be indexed.
- SummaryFld*..... An integer expression which provides the field number for the events table field (as specified by EventsTableNo) which will store the editable text summary for each event. The field's type must be **Alpha**.
- ColorFld*..... An integer expression which provides the field number for the events table field (as specified by EventsTableNo) which will store the color index or 24-bit color for each event.  
If the field's type is a Longint the field is to contain a 24-bit color. If the field's type is an Integer the field is to contain a color index. A color index is a number between 0 and 255 which identifies the color from the application's color palette.
- EventType*..... An integer expression which provides the field number for the events table field (as specified by EventsTableNo) which will store the Event Type for each event. The Event Type is a number with a value of either 0 or 1 which identifies a normal "timed" event (an integer value of 0) or a banner event (an integer value of 1). The field's type must be **Integer**.
- TextColorFld* ..... (Optional). An integer expression which provides the field number for the events table field (as specified by EventsTableNo) which will store the color index or 24-bit color for the text of each event. If the field's type is a Longint the field is to contain a 24-bit color. If the field's type is an Integer the field is to contain a color index. Pass a value of zero if this is not to be used.
- IconFld*..... (Optional). An integer expression which provides the field number for the events table field (as specified by EventsTableNo) which will store the icon

picture for each event, the field's type must be Picture. Pass a value of zero if this is not to be used.

**Error Codes:**

- 0 - Successful call made
- 15010 - Invalid text color field type
- 15011 - Invalid icon field type
- 15700 - Invalid SchedulePack Area
- 15001 - Invalid Event Table Number
- 15002 - Invalid Record ID Field Type
- 15003 - Invalid Event Summary Field Type
- 15004 - Invalid Start Date Field Type
- 15005 - Invalid End Date Field Type
- 15006 - Invalid Start Time Field Type
- 15007 - Invalid End Time Field Type
- 15008 - Invalid Color Field Type
- 15009 - Invalid Event Type Field Type
- 15010 – Invalid Text Color Field Type
- 15011 – Invalid Icon Picture Field Type

**Example:**

```
$X:=SXEventsTable(xSch;4;1;5;6;7;8;11;15;16;0;0) `Show table/field pointers here
```

To see an example, please refer to the [“Application Definition and Setup”](#) on page 86.

---

## SXLabelsTable

**Description:**

This function enables the developer to provide additional information about the application's database, needed when the developer wishes to take advantage of SchedulePack's resource scheduling features.

This routine is used prior to calling **SXShowDates** or **SXShowResources**, in order to enable the Resource labeling feature and identify the resource table and fields this feature is to use.

If the area object is not using the default events table, this routine must be called after calling **SXEventsTable**.

**Syntax:**

```
SXLabelsTable(xArea(L);EventLbIDFld(I);LbITableNo(I);LbITableIDFld(I);LbITableStrFld(I),  
LbITableColorFld)->L
```

**Parameters:**

*xArea*..... Pass either the value of a variable reference to a SchedulePack external-area object on a currently-active layout, or zero. If you pass a reference value, the command will apply only to that object. If you pass zero, the command will apply to the defaults for all future (as yet undisplayed) objects.

- EventLblIDFld* ..... An integer expression which provides the field number for the field in the current Events table which contains the key for the event's resource-table record. This field must be of type Long Integer.
- LblTableNo* ..... An integer expression which provides the table number for the database table in which the resource records are stored.
- LblTableIDFld* ..... An integer expression which provides the field number for the field in the resource table (as specified by *LblTableNo*) which stores the record ID for the resource records. This field must be of type Long Integer.
- LblTableStrFld* ..... An integer expression which provides the field number for the field in the resource table (as specified by *LblTableNo*) which stores the resource name. This field must be of type Alpha.
- LblTableColorFld* ..... An integer expression which provides the field number for the field in the resource table (as specified by *LblTableNo*) which shows the color index or 24-bit color for the resource records. If the field's type is a Longint the field is to contain a 24-bit color. If the field's type is an Integer the field is to contain a color index.
- LblTextColorFld* ..... An integer expression which provides the field number for the field in the resource table (as specified by *LblTableNo*) which shows the color index for the resource records. If the field's type is a Longint the field is to contain a 24-bit color. If the field's type is an Integer the field is to contain a color index.

**Error Codes:**

- 0 - Successful call made
- 15700 - Invalid SchedulePack Area
- 15100 - Invalid Label ID Field Type
- 15101 - Invalid Label Table Number
- 15102 - Invalid Label Table ID Field Type
- 15103 - Invalid Label Table Label Field Type
- 15104 - Invalid color field type
- 15105 - Invalid text color field type

**Example:**

```
$x:=SXLabelsTable(xArea;EventLblIDFld;LblTableNo;LblTableIDFld;LblTableStrFld;LblTableColorFld)
```

To see an example, please refer to the [“Application Definition and Setup”](#) on page 86.

---

## SXColumnsTable

**Description:**

This function enables the developer to provide additional information about the application's database, needed when the developer wishes to take advantage of SchedulePack's resource scheduling features.

Use this routine prior to calling **SXShowResources** to identify the table and fields for the resource group which comprises the desired column definitions for the grid.

Use this routine prior to calling **SXShowDates** to identify the table and fields for a resource group, when you wish to limit the display of events to those linked with a subset of the resources in that group.

If the area object is not using the default events table, this routine must be called after calling **SXEventsTable**.

**Syntax:**

**SXColumnsTable**(xArea(L);EventColIDFld(I);ColTableNo(I);ColTableIDFld(I);ColTableStrFld(I))->L

**Parameters:**

- xArea*..... Pass either the value of a variable reference to a SchedulePack external-area object on a currently-active layout, or zero. If you pass a reference value, the command will apply only to that object. If you pass zero, the command will apply to the defaults for all future (as yet undisplayed) objects.
- EventColIDFld*..... An integer expression which provides the field number for the field in the current Events table which contains the key for the event's resource-table record. This field must be of type Long Integer.
- ColTableNo*..... An integer expression which provides the table number for the database table in which the resource records are stored.
- ColTableIDFld*..... An integer expression which provides the field number for the field in the resource table (as specified by ColTableNo) which stores the record ID for the resource records. This field must be of type Long Integer.
- ColTableStrFld*..... An integer expression which provides the field number for the field in the resource table (as specified by ColTableNo) which stores the resource name. This field must be of type Alpha.

**Error Codes :**

- 0 - Successful call made
- 15700 - Invalid SchedulePack Area
- 15001 - Invalid Event Table Number
- 15200 - Invalid Column ID Field Type
- 15201 - Invalid Column Table number
- 15202 - Invalid Column Table ID Field Type
- 15203 - Invalid Column Table Label Field Type

**Example:**

**\$X:=SXColumnsTable**(xArea;EventColIDFld;ColTableNo;ColTableIDFld;ColTableStrFld)

To see an example, please refer to the [“Application Definition and Setup”](#) on page 86.

---

## SXSetView

**Description:**

Allows the developer to change the schedule area to a Resource, Days or Month view.

**Syntax:**

**SXSetView**(xArea(L);ViewKind(I)) ->L

**Parameters:**

*xArea*..... Pass either the value of a variable reference to a SchedulePack external-area object on a currently-active layout, or zero. If you pass a reference value, the command will apply only to that object. If you pass zero, the command will apply to the defaults for all future (as yet undisplayed) objects.

*View Kind*..... An integer expression indicating the kind of view to be shown.

1 - Resource or Dates View

2 - One-month View

3 - Time-line View

---

## SXGetView

**Description:**

Allows the developer to determine the schedule area type. Options are either a Resource, Days or Month view.

**Syntax:**

**SXGetView**(xArea(L);ViewKind(I)) ->L

**Parameters:**

*xArea*..... Pass either the value of a variable reference to a SchedulePack external-area object on a currently-active layout, or zero. If you pass a reference value, the command will apply only to that object. If you pass zero, the command will apply to the defaults for all future (as yet undisplayed) objects.

*View Kind*..... Returns an integer expression indicating the kind of view shown.

1 - Resource or Dates View

2 - One-month View

3 - Time-line View

**Example:**

**\$x:=SXGetView**(xSch;\$Viewkind)

## Utilities

---

### SXGetSelected

**Description:**

This function enables the developer to get the record ID of the event box currently selected by the user on the referenced SchedulePack area. The 4D developer can write procedures which use this record ID to search the events table and find the record corresponding to the selected event box.

Note that this function is only necessary when writing button scripts and menu procedures to support a given SchedulePack area on a layout; in a SchedulePack area's callback procedure, the developer can use the **SX\_PKey** callback variable instead.

**Syntax:**

**SXGetSelected**(xArea(L);RecordID(L))->L

**Parameters:**

- xArea*..... Pass the value of a variable reference to a SchedulePack external-area object on a currently-active layout.
- RecordID*..... A long integer value representing the value of the record ID field of the Events table record corresponding to the event box currently selected by the user. If no event box is selected, this value will be zero.

**Error Codes:**

- 0 - Successful call made
- 15700 - Invalid SchedulePack Area

**Example:**

\$err := **SXGetSelected**(xSch;\$recordID)

---

### SXDeleteEvent

**Description:**

This function enables the developer to delete a record in the application's events table, and to remove it from the display of the specified SchedulePack area.

**Syntax:**

**SXDeleteEvent**(xArea(L);RecordID(L))->L

**Parameters:**

- xArea*..... Pass the value of a variable reference to a SchedulePack external-area object on a currently-active layout.
- RecordID*..... A long integer expression which provides the record ID value of the record in the application's events table that the developer wishes to delete.

**Error Codes:**

- 0 - Successful call made
- 15700 - Invalid SchedulePack Area
- 15602 - Invalid Record ID Number

**Example:**

```
$RecordID:=SX_PKey  
$x:=SXDeleteEvent(xSch;$RecordID)
```

---

**SXColorPopup**

**Description:**

This function enables the 4D developer to make use of SchedulePack's color popup menu from within scripts of standard 4D layout objects. From the Runtime environment, the color popup menu is displayed when a User depresses the control-key (Mac/Windows) while clicking on the object whose color is to be changed. Once within the color pop-up, the User can direct the cursor to the desired color and release the mouse. Upon a keyboard "mouse-up", the color is automatically assigned the corresponding "Color ID" field within the associated event record.

When called, the routine will display a color popup menu, and will track the user's mouse as they select a color-item tile from the menu. When the user releases the mouse, the menu will close and the item number of the selected color-item will be returned.

The color numbers used by this routine correspond with the values used by 4D's SET COLOR command.

Note that, because the script that calls the routine has to be executed when the mouse button is pressed (and before it released), only those 4D layout objects that do so (such as an Invisible Button object) can be effectively used with this routine.

**Syntax:**

```
SXColorPopup( CurrentColor )->L
```

**Parameters:**

*CurrentColor* ..... An integer expression with a value of 0 to 255, which specifies the item number of the menu item that is to be highlighted as the current value when the menu is displayed.  
If the user fails to select a new color item (by moving the mouse completely off the menu before releasing the mouse button), this will be the value returned by the routine.

**Return Value:**

*Result* ..... An integer value representing the item-number of the color selected by the user.

**Example:**

```
$Result:= SXColorPopup(vColor)
```

## SXColorPicker

### Description:

This function displays the Color Picker dialog (Mac OS X only).

When called, the original color will be passed and the Macintosh OS color pickup dialog will appear with the color wheel showing a round marker designating the chosen color. After the end-user picks another color, the corresponding 24 bit valur for that color will be returned.

The color numbers used by this routine do not correspond with the values used by 4D's SET COLOR command.



Figure 2. Color Picker dialog

### Syntax:

```
SXColorPicker( CurrentColor(L);PickedColor(L)->L
```

### Parameters:

*CurrentColor* ..... A long integer, specifying the value of the current color.

*PickedColor* ..... Returns a long integer, specifying the value of the chosen color.

If the user fails to pick a new color item (by clicking the cancel button), the original value will be returned by the routine.

### Return Value:

*Result* ..... An integer valu: 0 = Cancel, 1 = Ok.

### Example:

```
$Result:= SXColorPicker(vCurrentColor; vPickedColor)
```

## SXColorToIndex

**Description:**

This function returns the 4D color palette number when given a 24bit color value.

**Syntax:**

```
SXColorToIndex( CurrentColor (L))->I
```

**Parameters:**

*24BitColor* ..... A long integer expression containing a 24-bit color value.

**Return Value:**

*Result* ..... An integer value from 0 to 255 if the 24-bit color is on the palette, -1 if is not.

**Example:**

```
$Result:= SXColorToIndex(v24BitColor)
```

---

## SXIndexToColor

**Description:**

This function returns a 24 bit color value given the 4D color palette index number.

**Syntax:**

```
SXIndexToColor(ColorIndex(I) )->L
```

**Parameters:**

*ColorIndex* ..... An integer expression with an index value from 0 to 255 of 4D's color palette.

**Return Value:**

*Result* ..... A long integer value which contains the 24-bit color value.

**Example:**

```
$colorValue:= SXIndexToColor(vIndexColor)
```

---

## SXColorToRGB

**Description:**

This is a function that returns the Red, Green and Blue color component values into integer variables when given a 24 bit color.

**Syntax:**

**SXColorToRGB**(24BitColor(L);RedComponent(I);GreenComponent(I);BlueComponent(I))

**Parameters:**

*24BitColor* ..... A long integer expression a 24-bit color value.

*RedComponent*..... Returns an integer value representing the red component of the color.  
A value from 0 thru 255.

*GreenComponent*..... Returns an integer value representing the green component of the color.  
A value from 0 thru 255.

*BlueComponent*..... Returns an integer value representing the blue component of the color.  
A value from 0 thru 255.

**Return Value:**

*Result* ..... Not applicable.

**Example:**

**SXColorToRGB**(v24BitColor; RedComponent; GreenComponent; BlueComponent)

---

## SXRGBToColor

**Description:**

This is a function that returns a 24 bit color value, in a long integer, when passed the Red, Green and Blue color component values.

**Syntax:**

**SXRGBToColor**( RedComponent(I); GreenComponent(I); BlueComponent(I))->L

**Parameters:**

*RedComponent*..... An integer value representing the red component. A value from 0 thru 255.

*GreenComponent*..... An integer value representing the green component. A value from 0 thru 255.

*BlueComponent*..... An integer value representing the blue component. A value from 0 thru 255.

**Return Value:**

*Result* ..... A long integer expression containing the 24 Bit Color value.

**Example:**

\$ColorValue:= **SXRGBToColor**(RedComponent; GreenComponent; BlueComponent)

---

## SXRowToTime

**Description:**

This function enables the developer to convert a row number into the corresponding time interval for a given SchedulePack area.

**Syntax:**

**SXRowToTime**(xArea(L);Row(I);Time(H))->L

**Parameters:**

*xArea*..... Pass the value of a variable reference to a SchedulePack external-area object on a currently-active layout.

*Row* ..... An integer expression which provides the row number of the SchedulePack external-area object on a currently-active layout.

*Time* ..... A time value representing the start of the time interval corresponding to the specified row number.

**Error Codes:**

0 - Successful call made  
 -15700 - Invalid SchedulePack Area  
 -15601 - Invalid Row Number

**Example:**

```
$Row:=1
$err:= SXRowToTime(xSch;$Row;$Time)
```

---

## SXTimeToRow

**Description:**

This routine enables the developer to convert a time interval into the corresponding row number for a given SchedulePack area.

**Syntax:**

**SXTimeToRow**(xArea(L);Time(H);Row(I))->L

**Parameters:**

*xArea*..... Pass the value of a variable reference to a SchedulePack external-area object on a currently-active layout.

*Time* ..... A time value representing the start of the time interval corresponding to the specified row number.

*Row* ..... An integer value representing the row number of the time interval on the grid containing the specified time. If the specified time value is not on the grid, a value of -1 is returned.

**Error Codes:**

- 0 - Successful call made
- 15700 - Invalid SchedulePack Area
- 15401 - Invalid Time

**Example:**

```
$Time:=†09:00:00†  
$err:= SXTimeToRow(xSch;$Time;$Row)
```

---

**SXGetScroll**

**Description:**

This function enables the developer to “read” the current scroll settings of a given area, so that other areas of the layout can be updated to reflect the current left-most visible column, or that the scroll-settings can be restored using the **SXSetScroll** routine.

**Syntax:**

```
SXGetScroll(xArea(L);Row(I);Column(I))->L
```

**Parameters:**

- xArea*..... Pass the value of a variable reference to a SchedulePack external-area object on a currently-active layout.
- Row* ..... An integer variable which will return the row number of the upper-most visible row of the grid. A value of zero represents the banner events row at the top of the grid; the time interval rows are numbered starting at 1.
- Column*..... An integer variable which will return the column number of left-most visible column of the grid.

**Example:**

```
$X:=SXGetScroll(xSch;$vRow;$vCol)
```

---

**SXSetScroll**

**Description:**

This function enables the developer to scroll the grid so that a specific row and column are positioned in the upper-left corner of the visible area.

**Syntax:**

```
SXSetScroll(xArea(L);Row(I);Column(I))->L
```

**Parameters:**

- xArea*..... Pass the value of a variable reference to a SchedulePack external-area object on a currently-active layout.
- Row* ..... An integer expression which represents the row of the grid that the area

should be scrolled to.

*Column*..... An integer expression which provides the column number of the grid that the area should be scrolled to.

**Error Codes:**

- 0 - Successful call made
- 15700 - Invalid SchedulePack Area
- 15600 - Invalid Column Number
- 15601 - Invalid Row Number

**Example:**

```
$x:=SXSetScroll(xSch;$vRow;$vColumn)
```

---

**SXRefresh**

**Description:**

This function enables the developer to force a given area to completely recreate its current display, using its current size and parameter settings and the currently applicable records in the database's events, column-resources and label-resources tables.

This function is usually only needed in a multi-user environment, when the current user has a SchedulePack area open while other users may be editing, adding or deleting records in the events table that may be relevant to the current user. By making judicious use of this function, the developer can make sure that the user's display contains up-to-date information.

This function performs the same functions as the user's Command-Enter keyboard command - it recalculates the optimum column widths and row heights, reloads the list of events to display in the grid, and redraws the display.

**Syntax:**

```
SXRefresh(xArea(L))->L
```

**Parameters:**

*xArea*..... Pass the value of a variable reference to a SchedulePack external-area object on a currently-active layout.

**Error Codes:**

- 0 - Successful call made
- 15700 - Invalid SchedulePack Area

**Example:**

```
$x:=SXRefresh(xSch)
```

## SXGetDraggedEvent

### Description:

This command returns the record ID of the dragged event. Call this command when the "dropped on" object receives an "On Drop" event and the source object is a SchedulePack area.

### Syntax:

**SXGetDraggedEvent**(xArea(L))->L

### Parameters:

*xArea*..... Pass the value of a variable reference to a SchedulePack external-area object on a currently-active layout.

### Error Codes:

0 - Successful call made  
-15700 - Invalid SchedulePack Area

### Example:

**\$RecordID:=SXGetDraggedEvent**(xSch)

---

## SXGetDroppedObject

### Description:

**SXGetDroppedObject** returns the "source" information of a drop onto a SchedulePack area. Call this when the SchedulePack callback method receives a "SxkOnDrop" action. Parameters 2,3,4 are the same as the corresponding parameter's in 4D's "DRAG AND DROP PRPERTIES" command.

### Syntax:

**SXGetDroppedObject**(xArea(L);SourcePtr(Z);SourceElement(L); SourceProcess(L); DestDate(D); DestResourceID(L); DestStartTime(H); DestEndTime(H))->L

### Parameters:

*xArea*..... Pass the value of a variable reference to a SchedulePack external-area object on a currently-active layout.

*SourcePtr* ..... Returns a Pointer to the source object.

*SourceElement* ..... Returns the array element of the dropped Object.

*SourceProcess*..... Returns the originating process where the dropped object came from.

*DestDate* ..... Returns a Date value (valid for either the Dates View, Months View, or Time-line View) of the drop location.

*DestResourceID*..... Returns a long integer value, specifying the Resource ID (only applicable in the Resources view where column headers are resources and not dates), and also the Time-line view of the drop location.

*DestStartTime* ..... Returns a Time value indicating the start time of the drop location.

*DestEndTime*..... Returns a Time value indicating the end time of the drop location.

**Error Codes:**

- 0 - Successful call made
- 15700 - Invalid SchedulePack Area

**Example:**

```
$x:=SXGetDroppedObject(xSch;vSrcPtr;vSourceElement;vSourceProcess;vDestDate;vDestResID;  
vDestStartTime;vDestEndTime)
```

**SXExport****Description:**

This command exports the current events from the specified SchedulePack area and exports them as an iCalendar formatted file. The time zone parameters are based on the iCalendar specification. When exporting, the end-user will be prompted to name the file. Instruct them to append the file name with the “.ics” suffix.

To import into iCal, select the ”Import” option from the iCal File menu and then select “Import an iCal File” when prompted (see below):

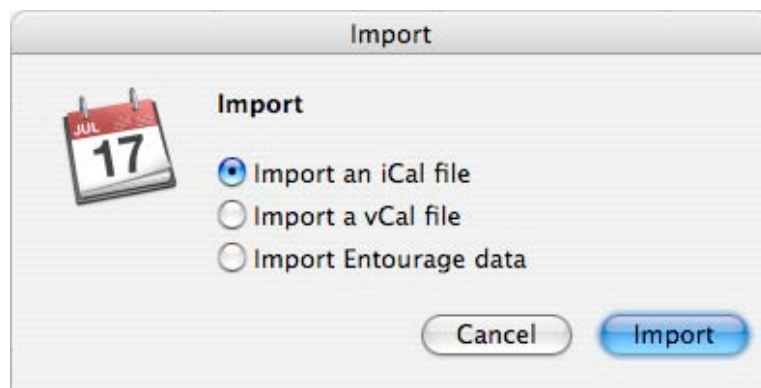


Figure 3. Import an iCal File

Next, the end user will be prompted to select the calendar where the activity data will be imported into. If the default “New Calendar” option is chosen, the calendar name will assume the same file name as the import file.

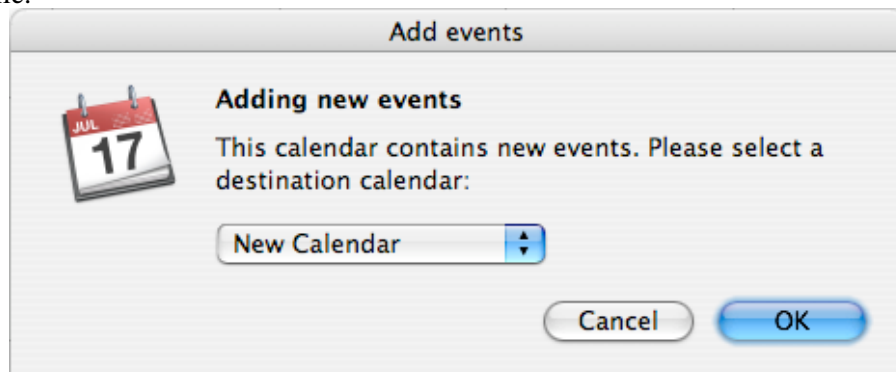


Figure 4. Add events

**Syntax:**

**SXExport**(xArea(L);TimeZoneKind(I);TimeZoneValue(S);DestFilePath(T))->L

**Parameters:**

- xArea*..... Pass the value of a variable reference to a SchedulePack external-area object on a currently-active layout.
- TimeZoneKind*..... An integer expression where 1 = TZID, 2 = TZ Name, 3 = TZ Offset From, 4 = TZ Offset To, 5 = TZ URL
- TimeZoneValue* ..... A string containing an appropriate value for the TimeZoneKind.
- DestFilePath* ..... Destination File Path

**Error Codes:**

- 0 - Successful call made
- 15700 - Invalid SchedulePack Area
- 15701 - Invalid time zone kind
- 15711 - Invalid file path
- 15712 - Could not open file for writing

**Example:**

**\$x:=SXExport**(xSch;vTZ\_Kind;TZ\_Value;vDest\_FilePath)

---

## SXImport

**Description:**

This command imports events from an iCalendar formatted file.

**Syntax:**

**SXImport** (xArea(L);MinutesOffset(L);SourceFilePath(T))->L

**Parameters:**

- xArea*..... Pass the value of a variable reference to a SchedulePack external-area object on a currently-active layout.
- MinutesOffset*..... A long integer expression which is the number of minutes to offset the event times from the iCalendar file.
- SourceFilePath* ..... A text value which specifies the file path of the iCalendar file.

**Error Codes:**

- 0 - Successful call made
- 15700 - Invalid SchedulePack Area
- 15711 - Invalid file path
- 15712 - Cannot open file for reading

**Example:**

**\$x:= SXImport** (xSch;120;\$FilePath)

## Calendar Control Routines

### The SchedulePack Calendar Area

SchedulePack v4 contains an additional 4D plug-in area which displays a monthly calendar. This object can be used to provide users with a convenient way to specify a date value. It can also be used to graphically summarize event information for a given month. The date cells can be embellished with up to three icons (SICNs); these icons as well as the date label for the cell can be drawn in independently specified colors. Event information for the month can be summarized in the display using symbolic and color-coded conventions.

To use the object, draw an external-area object on a layout and assign the name **%SXCalendar** as the object's external procedure. Make the object any size you wish, and the header areas and date cells will be automatically sized to fill the area. By default, the calendar will bevel the current day to distinguish it from all other days in the current month.

When the area is displayed to the user, they may select any date in the currently displayed month simply by clicking on it. The calendar will always have exactly one date selected. The user may scroll to the next or previous month by clicking on the arrow buttons in the upper-right or upper-left corner of the area. If the user holds down the Command key when clicking on these buttons, the calendar will scroll to the next or previous year.

Scrolling the calendar does not change the selected date - the user must click on a date cell in order to do so. If the currently selected date is in a month other than the one currently displayed in the area, the user may “jump” to the month of the currently selected date by clicking on the title-bar area of the calendar (between the two scroll-buttons). If the user holds down the Command key and clicks on the title-bar area, the calendar will select and display today's date.

All of the calendar area's features can be controlled procedurally using the routines provided by SchedulePack ([see “](#)

4D Developer ” on page 5 for detailed information).

The routines in this section enable the developer to interact with the **%SXCalendar** external area, to configure it for display or to determine its current date selection.

---

## SXCalFormat

### Description:

This routine enables the developer to assign the fonts used for the header and date cells of the calendar.

### Syntax:

**SXCalFormat**(xArea(L);Dayformat(I);Monthformat(I))

### Parameters:

*xArea*..... Pass either the value of a variable reference to a SchedulePack external-area object on a currently-active layout, or zero. If you pass a reference value, the command will apply only to that object. If you pass zero, the command will apply to the defaults for all future (as yet undisplayed) objects.

*Dayformat* ..... 1= Short format (M, T, W Not valid in a two byte system) 2=Abbreviated format (Mon, Tue, Wed + year) 3=Long format (Monday, Tuesday, etc. + year).

*MonthFormat* ..... 1= Short format (not valid, SchedulePack will use 2 if 1 is used) 2=Abbreviated format (Jan, Feb, Mar) 3=Long format

**Example:**

**SXCalFormat**(xSch;1;1)

---

**SXCalFonts**

**Description:**

This routine enables the developer to assign the fonts used for the header and date cells of the calendar.

**Syntax:**

**SXCalFonts**(xArea(L);HeaderFont(S);HeaderFontSize(I);HeaderFontStyle(I);CellFont(S);CellFontSize(I);CellFontStyle(I))

**Parameters:**

*xArea* ..... Pass either the value of a variable reference to a SchedulePack external-area object on a currently-active layout, or zero. If you pass a reference value, the command will apply only to that object. If you pass zero, the command will apply to the defaults for all future (as yet undisplayed) objects.

*HeaderFont* ..... A string expression specifying the name of the font to be used for the header areas (the month/year and weekday titles).

*HeaderFontSize* ..... An integer expression specifying the font size to be used for the header areas.

*HeaderFontStyle* ..... An integer expression specifying the font styles (encoded using the same values as 4D's FONT STYLE command) to be used for the header areas.

*CellFont* ..... A string expression specifying the name of the font to be used for the date cells.

*CellFontSize* ..... An integer expression specifying the font size to be used for the date cells.

*CellFontStyle* ..... An integer expression specifying the font styles to be used for the date cells.

**Example:**

**SXCalFonts**(xSch;"Geneva";12;2;"Geneva";10;1)

---

**SXCalDateColor**

**Description:**

This function enables the developer to assign a text color to be used for the date number in a specific date cell.

**Syntax:**

**SXCalDateColor**(xArea(L);DayNumber(I);TextColor(I))->L

**Parameters:**

- xArea*..... Pass the value of a variable reference to a SchedulePack external-area object on a currently active layout.
- DayNumber*..... An integer expression which specifies the desired date of the cell for the currently displayed month.
- TextColor*..... An integer expression which specifies the index to the desired color in the current color palette.

**Error Codes:**

- 0 - Successful call made  
-15700 - Invalid SchedulePack Area  
-15403 - Invalid Day Number

**Example:**

**\$x:=SXCalDateColor**(xSch;1;2) 'Make first day of month yellow

To see an example, please refer to the [“SchedulePack Area”](#) on page 90.

---

## SXCalDateIcon

**Description:**

This function enables the developer to display up to three small icons (SICNs) in the date cells of the calendar. The icon can be chosen from the list of icons stored in the SICN resource in the SchedulePack table resource fork. This list can be extended by editing the SICN using any Macintosh resource editor such as ResEdit or Resorcerer.

**Syntax:**

**SXCalDateIcon**(xArea(L);DayNumber(I);IconLocation(I);IconIndex(I);IconColor(I))->L

**Parameters:**

- xArea*..... Pass the value of a variable reference to a SchedulePack external-area object on a currently-active layout.
- DayNumber*..... An integer expression which specifies the desired date of the cell for the currently-displayed month.
- IconLocation*..... An integer expression which specifies the desired relative location for the icon in the date cell. The following coded values are valid:
- |                 |                        |
|-----------------|------------------------|
| Value Location: | 1 - Upper-right corner |
|                 | 2 - Lower-left corner  |
|                 | 3 - Lower-right corner |
- IconIndex*..... An integer expression which specifies the index of the desired icon in the SchedulePack SICN resource.

*IconColor*..... An integer expression which specifies the index to the color in the current color palette to use to draw the icon.

**Error Codes:**

- 0 - Successful call made
- 15700 - Invalid SchedulePack Area
- 15403 - Invalid Day Number
- 15500 - Invalid Icon Index

**Example:**

`$x:=SXCalDateIcon(xSch;1;1;7;2)` Places SICN #7 on the first day of the month in the upper-right corner and color it yellow.

To see an example, please refer to the “[SchedulePack Area](#)” on page 90.

---

### SXCalClear

**Description:**

This function enables the developer to clear the effect of all previous calls to **SXCalDateColor** and **SXCalDateIcon**. Since the effects of these routines are cumulative and persistent, calling this function is generally required as the first step when configuring a given month's calendar.

**Syntax:**

`SXCalClear(xArea(L))->L`

**Parameters:**

*xArea*..... Pass the value of a variable reference to a SchedulePack external-area object on a currently-active layout.

**Error Codes:**

- 0 - Successful call made
- 15700 - Invalid SchedulePack Area

**Example:**

`$x:=SXCalClear(xSch)`

To see an example, please refer to the “[SchedulePack Area](#)” on page 90.

---

### SXCalGetMonth

**Description:**

This function enables the developer to retrieve the number (1-12) of the month currently displayed in a calendar area.

**Syntax:**

`SXCalGetMonth(xArea(L);CurrentMonth(l))->L`

### Parameters:

- xArea*..... Pass the value of a variable reference to a Schedule external-area object on a currently-active layout.
- CurrentMonth* ..... An integer value from 1 to 12 specifying the calendar month currently displayed.

### Error Codes:

- 0 - Successful call made  
-15700 - Invalid SchedulePack Area

### Example:

`$x:=SXCalGetMonth(xSch;vMonth)`

To see an example, please refer to the “SchedulePack Area ” on page 90.

---

## SXCalGetYear

### Description:

This routine enables the developer to retrieve the year currently displayed in a calendar area.

### Syntax:

`SXCalGetYear(xArea(L);CurrentYear(L))->L`

### Parameters:

- xArea*..... Pass the value of a variable reference to a SchedulePack external-area object on a currently-active layout.
- CurrentYear* ..... A long integer value specifying the calendar year currently displayed.

### Error Codes:

- 0 - Successful call made  
-15700 - Invalid SchedulePack Area

### Example:

`$x:=SXCalGetYear(xSch;vYear)`

To see an example, please refer to the “SchedulePack Area ” on page 90.

---

## SXCalSetMonthYr

### Description:

This routine enables the developer to display the calendar for a specific month and year. Note that changing the month displayed in a calendar does not change its selected date - the user will have to click on a date in the currently displayed month to do that. Alternatively, the **SXCalSet-Date** routine can be used.

**Syntax:**

**SXCalSetMonthYr**(xArea(L);Month(I);Year(L))

**Parameters:**

- xArea*..... Pass the value of a variable reference to a SchedulePack external-area object on a currently-active layout.
- Month*..... An integer value between 1 and 12 specifying the desired month of the year to display.
- Year*..... An integer value specifying the calendar year to display.

**Example:**

**SXCalSetMonthYr**(xSch;1;2010) `Display the first month of 2010

---

**SXCalGetDate**

**Description:**

This function enables the developer to determine the calendar's currently selected date.

**Syntax:**

**SXCalGetDate**(xArea(L);SelectedDate(D))->L

**Parameters:**

- xArea*..... Pass the value of a variable reference to a SchedulePack external-area object on a currently-active layout.
- SelectedDate* ..... A date value which represents the date of the cell currently selected by the user.

**Error Codes:**

- 0 - Successful call made
- 15700 - Invalid SchedulePack Area

**Example:**

\$x:=**SXCalGetDate**(xSch;vSelectDate)

## SXCalSetDate

### Description:

This routine enables the developer to select a specific date in the calendar. Note that selecting a date does not automatically cause it to be displayed - use the **SXCalSetMonthYr** routine to display a given month and year in the calendar.

### Syntax:

**SXCalSetDate**(xArea(L);DateToSelect(D))

### Parameters:

*xArea*..... Pass the value of a variable reference to a SchedulePack external-area object on a currently-active layout.

*DateToSelect*..... A date expression specifying the date to select in the calendar.

### Example:

\$CurDate:=Current Date

**SXCalSetDate**(xArea;\$CurDate)

To see an example, please refer to the “SchedulePack Area ” on page 90.

## Month View Commands

October 2009						
Sunday	Monday	Tuesday	Wednesday	Thursday	Friday	Saturday
27	28	29	30	1	2	3
				2 Day Therapy session		Eng201 at 9 AM on 9/13/9
4	5	6	7	8	9	10
• Hey There	• Meeting in room 2	Phoenix Seminar (2 days) he • Hi There!	4 Day layover • New Red October Event	• Therapist 5 • This is event #1	Hello, Hello • Hapi • Bacl • Ther • There • Hydi • Roal • ID#2	• New record • Here we go • Appointment • Monday the
11	12	13	14	15	16	17
• Therapist # • Morning ses • 1 Hour and • Open house • 45 minutes	Weeklong forest retreat • Head 120 C • Here we gro • 1 Hour & 5C • Morning ses	• Duplicate Event	• Why n • 9-10 C • New Co • Read1 • 30 min	• 9-10 Color • True Blue • I L like this : • 60 minutes	Facial wrap should be wrapped a few times. • MA101 • 120 MI • 15 min • MA101 • 75 min • Imports	• Room 3; 2-3PM
18	19	20	21	22	23	24
It's a jungle out here • Relax in the pool... Why r • Morning Session Appoint	Super Long Weekend event • 1 Hour 15 minutes	• Hi There! • Toby came l • Hey there!	Long weekend session • Lobotomy is next on our l		• Geography • Hi There! In • summary r • Training ses • nothing	Annual Dallas Conference • Back to the Future • Testing and Evaluation!
25	26	27	28	29	30	31
• How did it happen? Where • Where is it? How did it ha	2 day intensive • Dunder Mifflin Meeting	• Duplicate Event • test 1		9-10 Color ID=200 • Room 5	• Big Blue Ev • Hydrothera • Therapist #	• Another mo • Hi There! • 2 Hours for • hello there

## The SchedulePack Month View

This command sets general month view options. SchedulePack provides a multi-purpose monthly view providing a wide scopy of capabilities and flexibility for applying events in a familiar monthly area.

---

### SXMonthOpt

**Description:**

Allows the developer to set the Month view options.

**Syntax:**

**SXMonthOpt**(xArea(L); StartDayOfWeek (I); ShowNonMonthDays (I);EventOrder(I)) ->L

**Parameters:**

- xArea*..... Pass the value of a variable reference to a SchedulePack external-area object on a currently-active layout.
- StartDayOfWeek* ..... An integer expression indicating the day of week.
  - 1 - Sunday
  - 2 - Monday
  - 3 - Tuesday
  - 4 - Wednesday
  - 5 - Thursday
  - 6 - Friday
  - 7 - Saturday
  - 1 - no change
- ShowNonMonthDays*..... An integer expression indicating whether a month area will display days that precede and follow the current month (i.e., the month immediately before and the month immediately after).
  - 0 - No
  - 1 - Yes
  - 1 - no change
- EventOrder*..... An integer expression indicating the order of how events are drawn.
  - 1 - Left to right, then top to bottom.
  - 2 -Top to bottom, then left to right.
  - 1 - no change.

**Example:**

**\$x:=SXMonthOpt**(xSch;\$WeekStartDay;\$ShowNonMonthDays;\$EventOrder)

---

### SXMonthFont

**Description:**

Allows the developer to set the font information for the various Month view parts.

**Syntax:**

**SXMonthFont**(xArea(L);Part(I);Font Name(S);Font Size(I);Font Style(I);Font Color(I)) ->L

**Parameters:**

- xArea*..... Pass the value of a variable reference to a SchedulePack external-area object on a currently-active layout.
- Part* ..... An integer expression indicating the day of week.
  - 1 - Month & Year
  - 2 - Weekday Names
  - 3 - Day Numbers
  - 4 - non-month day numbers
- Font Name*..... A string expression providing a valid font name. Empty string for no change.
- Font Size* ..... An integer expression providing a valid font size. -1 for no change.
- Font Style* ..... An integer expression providing a valid font style. -1 for no change.
- Font Color*..... A long integer expression containing a valid font color. -1 for no change.

**Example:**

**\$x:=SXMonthFont**(xSch;vPart;vFontName;vFontSize;vFontStyle;vFontColor)

---

## SXMonthClearPictures

**Description:**

Removes pictures from the date range for the month view.

**Syntax:**

**SXMonthClearPictures**(xArea(L);FromDate(D);ToDate(D)) ->L

**Parameters:**

- xArea*..... Pass the value of a variable reference to a SchedulePack external-area object on a currently-active layout.
- FromDate*..... A date expression indicating the start day from which pictures will be removed.
- ToDate* ..... A date expression indicating the last day from which pictures will be removed.

**Example:**

**\$x:=SXMonthClearPictures**(xSch;vFromDate;vToDate)

### SXMonthGetMonthYear

**Description:**

Returns month and year shown for the current Month view.

**Syntax:**

**SXMonthGetMonthYear**(xArea(L);Month(I);Year(I)) ->L

**Parameters:**

- xArea*..... Pass the value of a variable reference to a SchedulePack external-area object on a currently-active layout.
- Month*..... Returns an integer value, ranging from 1 to 12, indicating the current month on the “Month” view.  
1 - January, 2 - February, 3- March,  
4- April, 5 - May, 6- June, 7- July,  
8- August, 9 - September, 10 - October,  
11 - November & 12 - December
- Year*..... Returns an integer value indicating the current year (4 digits) on the “Month” view.

**Example:**

**\$x:=SXMonthGetMonthYear**(xSch;vMonth;vYear)

---

### SXMonthSetColor

**Description:**

Sets the color for a specific part of the Month view.

**Syntax:**

**SXMonthSetColor**(xArea(L);Part(I);Color(L)) ->L

**Parameters:**

- xArea*..... Pass the value of a variable reference to a SchedulePack external-area object on a currently-active layout.
- Part* ..... An integer expression, specifying a specific part of the Month View as follows:  
1 = Background, 2 = Days, 3 = Border of Days, 4 = Non-month days, 5 = Border of non-month days.
- Color*..... A long integer expression containing a 24-bit color value to be applied to the specified part.

**Example:**

**\$x:=SXMonthSetColor**(xSch;vPart;vColor)

### SXMonthSetDatePicture

**Description:**

Sets the picture to be drawn on a specific date when the view is the Month view.

**Syntax:**

**SXMonthSetDatePicture**(xArea(L);Date(D);Picture(P);HorizLoc(I);VerticalLoc(I);SizeType(I);MaxSize(I)) ->L

**Parameters:**

- xArea*..... Pass the value of a variable reference to a SchedulePack external-area object on a currently-active layout.
- Date*..... A date expression, indicating the date where the associated picture (for this command) will be drawn.
- Picture*..... A picture that is to be drawn on one specific date within the month view
- HorizLoc* ..... An integer value where 1 = left, 2 = center & 3 = right.
- VerticalLoc*..... An integer value where 1 = top, 2 = center & 3 = bottom.
- SizeType* ..... An integer value where 1 = fixed size & 2 = scaled.
- MaxSize*..... An integer value reflecting the maximum size if SizeType = 1 or a percentage of date if SizeTye = 2.

**Example:**

**\$x:=SXMonthSetDatePicture**(xSch;vDate;vPicture;vHorizLoc;vVerticalLocvSizeType;vMaxSize)

---

### SXMonthSetDayColor

**Description:**

Sets the color for a specific day within the current Month view.

**Syntax:**

**SXMonthSetDayColor**(xArea(L);DayNumber(I);Color(L)) ->L

**Parameters:**

- xArea*..... Pass the value of a variable reference to a SchedulePack external-area object on a currently-active layout.
- DayNumber*..... An integer expression, where 0 = current day and 1 – 31 represent the range of possible days.
- Color*..... A long integer specifying a 24-bit color value.

**Example:**

**\$x:=SXMonthSetDayColor**(xSch;vDayNumber;;vColor)

### SXMonthSetMonthYear

**Description:**

Sets the month (and year) to be displayed within the current Month view.

**Syntax:**

**SXMonthSetMonthYear**(xArea(L);Month(I);Year(I)) ->L

**Parameters:**

*xArea*..... Pass the value of a variable reference to a SchedulePack external-area object on a currently-active layout.

*Month*..... An integer, from 1 to 12, representing the month to be shown.

*Year*..... An integer indicating the year (in 4 digits) associated with the month specified.

**Example:**

**\$x:=SXMonthSetMonthYear**(xSch;vMonth;;vYear)

## Timeline Commands



Figure 5. Timeline Commands

## The SchedulePack Timeline Commands

**Description:**

SchedulePack 4 contains a new timeline view, engineered to proactively and visually solve a variety of potential multi-resource scheduling conflicts. This timeline area provides interactive, multi-day

time-line viewing and editing capabilities. Flexible, it works with either 4D records or arrays (via the callback), at the developer's discretion. The Timeline provides users a dynamic overview of daily events, including full print capabilities, for any number of resources (Users, rooms, equipment, classes, etc.). In addition to the SchedulePack Resource, Date and Monthly Views, the SXEventFont, SXAreaOpt, SXTimeIntervals, SXEventsTable, SXColumnsTable and the SXShowDates commands also apply to the timeline area. When developing a multi-view calendar area, these central commands can allow you to easily standardize the views attributes, when appropriate.

---

## SXTimeLineOpt

### Description:

This function specifies the column width and header height, the resource header width and height, the column, resource and time grid lines size for the Timeline view.

### Syntax:

```
SXTimeLineOpt(xArea(L);ColumnWidth(I);ColumnHeaderHeight(I);ResourceHeaderWidth(I);ResourceHeight(I);  
ColumnGridLinesSize(I);ResourceGridLinesSize(I);TimeGridLinesSize(I))->L
```

### Parameters:

- xArea*..... Pass the value of a variable reference to a SchedulePack external-area object on a currently-active layout.
- ColumnWidth* ..... An integer expression representing the column width preference (-1 = No Change, 0 = auto, 1 = scale to fit, >5 = fixed points).
- ColumnHdrHeight* ..... An integer expression representing the column header height preference (-1 = No Change, 0 = auto, >5 = fixed points).
- ResourceHdrHt* ..... An integer expression representing the resource header width preference (-1 = No Change, 0 = auto, >5 = fixed points).
- ResourceHeight*..... An integer expression representing the resource height preference (-1= No Change, 0 = auto, 1 = scale to fit, >5 = fixed points).
- ColumnGridLinesSize* ..... An integer expression representing the column grid lines size preference (-1 = No Change, 0 = hidden, >0 = fixed points).
- ResourceGridLinesSize* ..... An integer expression representing the resource grid lines size preference (-1 = No Change, 0 = hidden, >0 = fixed points).
- TimeGridLinesSize*..... An integer expression representing the time grid lines size preference (-1 = No Change, 0 = hidden, >0 = fixed points).

### Example:

```
$x:= SXTimeLineOpt(xSch;ColumnWidth;ColumnHeaderHeight;ResourceHeaderWidth;ResourceHeight;  
ColumnGridLinesSize;ResourceGridLinesSize;TimeGridLinesSize)
```

---

## SXTimeLineFont

### Description:

This function changes font characteristics for the Column Headers or the Reource Headers.

**Syntax:**

**SXTimeLineFont**(xArea(L);Part(I);Fontname(S); Fontsize(I); Fontstyle(I);FontColor(L)->L

**Parameters:**

- xArea*..... Pass the value of a variable reference to a SchedulePack external-area object on a currently-active layout.
- Part* ..... An integer expression representing either the Column or the Resource headers (1 = Column headers and 2 = Resource headers).
- Fontname* ..... A string expression representing the font name (“” = no change, “Tahoma”, “Geneva”, etc. equals the font name).
- Fontsize* ..... An integer expression representing the font size (-1 = no change, n = font size (from 4 to 80).
- Fontstyle*..... An integer expression representing the font style (-1= no change, 0 = Plain, 1 = Bold, 2 = Italic, 4 = Underline, 8 = Outline, 16 = Shadow, 32 = Condensed, 64 = Extended
- FontColor*..... A long integer expression containing a 24-bit color value. (-1 = no change).

**Example:**

**\$x:=SXTimeLineFont**(xSch;1;"Helvetica";9;0;\$FontColor)

---

**SXTimeLineSetColor**

**Description:**

This function changes the color of the Column, Resource and Grid Headers background and the Column, Resource and Grid lines.

**Syntax:**

**SXTimeLineSetColor**(xArea(L);Part(I);Color(L)->L

**Parameters:**

- xArea*..... Pass either a long integer value of a variable reference to the chart external-area object on a currently-active layout, or zero. If you pass a reference value, the command will apply only to that object. If you pass zero, the command will apply to the defaults for all future (as yet undisplayed) objects.
- Part* ..... An integer expression representing the following areas (1 = Column headers background, 2 = Resource headers background, 3 = Grid Background, 4 = esource Grid Lines, 5 = Column Grid Lines, 6 = Time Grid Lines).
- Color* ..... A long integer expression containing a 24-bit color value. (-1 = no change).

**Example:**

**\$x:=SXTimeLineSetColor**(xSch;3;\$Color)

## Chart Commands

### The SchedulePack Chart Commands

The SchedulePack chart commands provide developers a convenient array-based tool for viewing and printing daily time allocations for a specific resource (a user, room, piece of equipment, class, etc.).

This chart area is ideal for multiple events for a single resource for a specific day. While the Chart plug-in area is not interactive, it can be used to graphically summarize daily event information. Each chart or “grid” has column headers providing developer defined text labels. The text size, font and color can be customized as well as the color of the underlying background area for the entire column header area. Secondly, each chart has 1 or more rows headers, also providing developer defined text labels along with an underlying background area. Lastly, each chart area has a grid for object display, including a developer definable background area and vertical and horizontal gridlines to separate each time slice. Developers can provide direct chart or time-line printing of whatever they can display. Presently, there is no editing or callback available within the chart area of SchedulePack.

There are seven related charting commands used for display purposes.

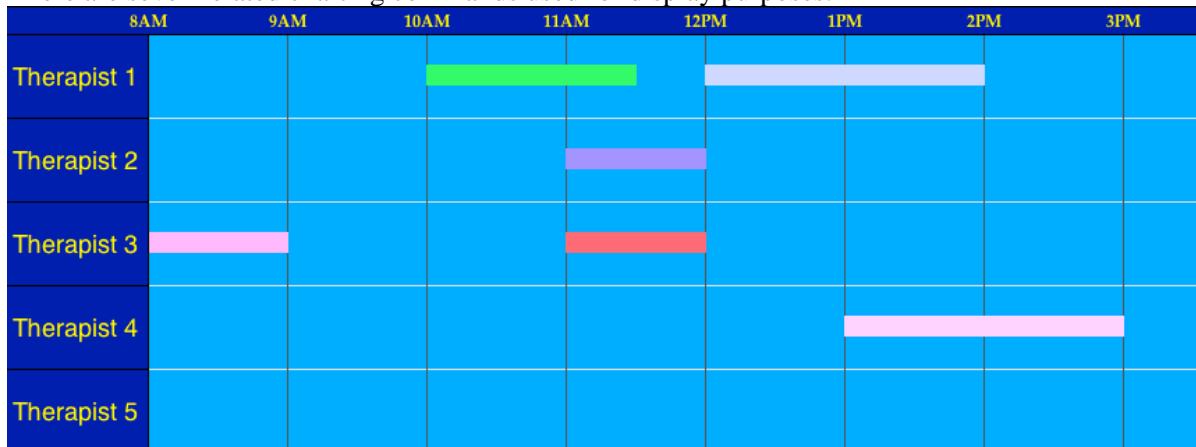


Figure 6. SchedulePack Chart Commands

The SXkQueryBefore callback capability allows 4D developers full control over the SchedulePack Chart display area. The command is useful for populating the SchedulePack arrays according to specific requirements. In addition, the SXkQueryBefore command allows developers to generate and display non-database related data directly into the SchedulePack area. This has a number of viable uses, such as; blocking out lunch or down-time blocks with specific color-coded events, marking a scheduling area with important non-database related events or displaying down times for specific resource related events. Whatever the challenge, providing 4D developers the flexibility to populate a scheduling area according to their unique demands opens up exciting possibilities for any variety of scheduling challenges.

With the charting tool, overlapping event objects can be generously spaced when scheduling conflicts exist.

Developers can now control the precision of overlapping and accuracy. Improved display for overlapping events is now included in both the SchedulePack area and the new Chart area.

The SXTimeIntervals command accepts both time and integer values, allowing the SchedulePack Chart plug-in area to draw and extend beyond the midnight limitation imposed under earlier versions. This

provides the time display columns the ability to extend after midnight into the following day. This is valuable for organizations needing to manage scheduling from one day over into the next.

## SXChartColumns

**Description:**

This function is for use with the Chart plug-in area and specifies the Chart area., the row header and the individual row labels.

**Syntax:**

**SXChartColumns(L;I;X;L;L)**

**SXChartColumns**(xArea(L);Alignment(I);Column Header(s)(Array of Text);Start Value(s)(Array of Long Integers);End Value(s)(Array of Long Integers))

**Parameters:**

- xArea*..... Pass either a long integer value of a variable reference to the chart external-area object on a currently-active layout, or zero. If you pass a reference value, the command will apply only to that object. If you pass zero, the command will apply to the defaults for all future (as yet undisplayed) objects.
- Alignment*..... An integer expression relating to the Header alignment (0 = center over divider, 1 = center in column).
- Column Headers*..... An array of text containing the column headers.
- Start Value*..... An array of long integers expressing the starting times for a particular row.
- End Value*..... An array of long integers expressing the ending times for a particular row.

**Example:**

**\$!\_Error:=SXChartColumns** (\$ChartArea;0;at\_Column\_Headers;al\_StartTime;al\_EndTime)

## SXChartResources

**Description:**

This function is for use with the Chart plug-in area and specifies the Chart area., the row header and the individual row labels.

**Syntax:**

**SXChartResources(L;T;X)**

**SXChartResources**(xArea(L);Row Header(T);Resources(Array of text))

**Parameters:**

- xArea*..... Pass either a long integer value of a variable reference to the chart external-area object on a currently-active layout, or zero. If you pass a reference value, the command will apply only to that object. If you pass

zero, the command will apply to the defaults for all future (as yet undisplayed) objects.

*Row Header* ..... A text expression specifying the row header.

*Resources* ..... An array of text labels identifying each of the row areas.

**Example:**

**\$!\_Error:=SXChartResources** (\$ChartArea;"Employees";aResources)

---

## SXChartSetResourceValues

**Description:**

This function enables the developer to specify the Chart plug-in's starting and ending times for each row as well as the color for the specific block of time.

**Syntax:**

**SXChartSetResourceValues(L;L;X;X;X)**

**SXChartSetResourceValues**(xArea(L);Resource Index(I);Start Time(s)(Array of Long Integer); End Time(s)(Array of Long Integer);Colors(Array of Integer))

**Parameters:**

*xArea* ..... Pass either a long integer value of a variable reference to the chart external-area object on a currently-active layout, or zero. If you pass a reference value, the command will apply only to that object. If you pass zero, the command will apply to the defaults for all future (as yet undisplayed) objects.

*Resource Index* ..... An integer expression specifying the row number.

*Start Times* ..... An array of long integers expressing the starting times for a particular row.

*End Times* ..... An array of long integers expressing the ending times for a particular row.

*Color* ..... An array of integers expressing the font colors for a particular row.

*Background Color* ..... An integer expression specifying the background color to be used for the column header (-1 = no change).

**Example:**

**\$!\_Error:=SXChartSetResourceValues**(\$ChartArea;\$i;aI\_StartTime;aI\_EndTime;ai\_Colors)

---

## SXChartColHdrOpt

**Description:**

This function enables the developer to specify detailed information for the Chart plug-in's column header area. Developers can specify column widths, header heights along with all associated font size, type and color information. In addition, the background color can be specified using the number associated with the 4D color chart.

**Syntax:**

**SXChartColHdrOpt(L;I;S;I;I;I)**

**SXChartColHdrOpt**(xArea(L);ColumnWidth(I);HeaderHeight(I);FontName(S);FontSize(I);Font-Style(I);FontColor(I);Background Color(I))

**Parameters:**

- xArea*..... Pass either a long integer value of a variable reference to the chart external-area object on a currently-active layout, or zero. If you pass a reference value, the command will apply only to that object. If you pass zero, the command will apply to the defaults for all future (as yet undisplayed) objects.
- ColumnWidth* ..... An integer expression specifying the width of the column (-1 = no change, 0 = auto, 1 = scale to fit, >5 = fixed).
- HeaderHeight*..... An integer expression specifying the font size to be used for the header areas (-1 = no change, 0 = auto, 1 = scale to fit, >5 = fixed).
- FontName*..... An string expression specifying the font name (a null value for no change).
- FontSize* ..... An integer expression specifying the font size (-1 = no change).
- FontStyle* ..... An integer expression specifying the font style (-1 = no change).
- FontColor*..... An integer expression specifying the font color (-1 = no change).
- Background Color*..... An integer expression specifying the background color to be used for the column header (-1 = no change).

**Example:**

**\$!\_Error:=SXChartColHdrOpt (\$ChartArea;100;0;"Palatino";12;1;1;5)**

**SXChartResourcesOpt**

**Description:**

This function enables the developer to establish label display options for the Chart plug-in's rows/resources.

**Syntax:**

**SXChartResourcesOpt(L;I;S;I;I;I)**

**SXChartResourcesOpt**(xArea(L);HeaderWidth(I);RowHeight(I);FontName(S);FontSize(I);Font-Style(I);FontColor(I);Background Color(I))

**Parameters:**

- xArea*..... Pass either a long integer value of a variable reference to the chart external-area object on a currently-active layout, or zero. If you pass a reference value, the command will apply only to that object. If you pass zero, the command will apply to the defaults for all future (as yet undisplayed) objects.
- Header Width* ..... An integer expression specifying the width of the column (-1 = no change, 0 = auto, >5 = fixed).

- RowHeight* ..... An integer expression specifying the font size to be used for the header areas (-1 = no change, 0 = auto, 1 = scale to fit, >5 = fixed).
- FontName*..... An string expression specifying the font name (a null value for no change).
- FontSize* ..... An integer expression specifying the font size (-1 = no change).
- FontStyle* ..... An integer expression specifying the font style (-1 = no change).
- FontColor*..... An integer expression specifying the font color (-1 = no change).
- Background Color*..... An integer expression specifying the background color to be used for the column header (-1 = no change).

**Example:**

```
$!_Error:=-SXChartResourcesOpt($ChartArea;0;60;"Helvetica";18;0;1;5)
```

---

## SXChartGridOpt

**Description:**

This function enables the developer to establish grid display options for the Chart plug-in area.

**Syntax:**

```
SXChartGridOpt(L;I;S;I;I;I)  
SXChartGridOpt(xArea(L);HorizontalLinesVisible(I);HorizontalLinesSize(I);HorizontalLinesColor(S);  
VerticalLinesVisible(I);VerticalLinesSize(I);VerticalLinesColor(S);Background Color(I))
```

**Parameters:**

- xArea*..... Pass either a long integer value of a variable reference to the chart external-area object on a currently-active layout, or zero. If you pass a reference value, the command will apply only to that object. If you pass zero, the command will apply to the defaults for all future (as yet undisplayed) objects.
- HorizontalLinesVisible* .....An integer expression specifying whether horizontal grid lines will be shown (-1 = no change, 0 = no, 1 = yes).
- HorizontalLinesSize* ..... An integer expression specifying the horizontal line size to be used for the grid (-1 = no change, or "n", where n is the number of pixels).
- HorizontalLinesColor* ..... An string expression specifying the 4D color palette # (-1 for no change).
- VerticalLinesVisible*..... An integer expression specifying whether vertical grid lines will be shown (1 = no change, 0 = no, 1 = yes).
- VerticalLinesSize* ..... An integer expression specifying the vertical line size to be used for the grid (-1 = no change, or "n", where n is the number of pixels).
- VerticalLinesColor*..... An string expression specifying the 4D color palette # (-1 for no change).
- Background Color*..... An integer expression specifying the background color to be used for the column header (-1 = no change).

**Example:**

```
I_Error:=-SXChartGridOpt($ChartArea;1;1;Light Grey;1;1;Dark Grey;Light Blue)
```

## SXChartAreaOpt

### Description:

This function is for showing/hiding scroll bars within the Chart plug-in area.

### Syntax:

**SXChartAreaOpts(L;I;I)**  
**SXChartResources**(xArea(L);Horizontal Scroll Bar(I);Vertical Scroll Bar(I))

### Parameters:

- xArea*..... Pass either a long integer value of a variable reference to the chart external-area object on a currently-active layout, or zero. If you pass a reference value, the command will apply only to that object. If you pass zero, the command will apply to the defaults for all future (as yet undisplayed) objects.
- Horizontal Scroll Bar*..... An integer expression specifying to show or hide the horizontal scroll bars (1 = no change, 0=hide, 1 =show).
- Vertical Scroll Bar*..... An integer expression specifying to show or hide the vertical scroll bars (-1 = no change, 0=hide, 1 =show).

### Example:

**\$!\_Error:=SXChartAreaOpts (\$ChartArea;1;\$1)'Show horizontal and vertical scroll bars**

## Callback Reference

### Overview

The 4D callback features allow the 4D developer to customize many of the operations that are otherwise automatically performed by the SchedulePack package. They enable the developer to extend or modify specific operations by simply writing a 4D procedure, which will be called by the package at key points in time — generally, whenever the user is performing a critical action through SchedulePack's graphical interface.

The callback feature enables the 4D developer to: [a] validate the action before it is allowed to occur, and [b] extend the action by adding application-specific processing in lieu of that which is automatically performed by SchedulePack.

Before each callback is executed, the SchedulePack package will initialize a set of 4D global variables with values that describe the event upon which the callback action is being performed. The reason global variables are used instead of standard procedure parameters is so that the procedure can modify the values as part of the validation process.

The callback procedure must declare a boolean field for the \$0 parameter “C\_Boolean(\$0)”. The developer will set \$0 equal to True when it wants SchedulePack to automatically perform its update. If the developer desires to perform all updates independent of SchedulePack, \$0 should be set to False.

Upon completion of the callback procedure, the SchedulePack package can apply the current values of these variables to describe the event, thereby automatically applying the changes made by the callback procedure.

### Callback Variables

The package will make use of one or more (usually more) of the following specially-named global variables for this purpose:

#### **SX\_PKey**

Formerly SX\_RecNo, this variable will be used to convey the unique ID value for the given event, corresponding to the record-ID field for the event's record in the database's events table.

When the edit-event or delete-event callback procedures are called, this variable will contain the ID value for an existing record in the events table. The callback procedure can use this value to retrieve the corresponding record and access other application-defined fields.

When the new-event callback procedure is called, this variable will initially contain a zero value, since the record has not yet been saved in the database. If the application has its own mechanism for deriving and assigning longInt keys to records, or if the application requires records in other tables to be related to new event records, the callback procedure may derive a key value and assign it to this variable at this time. In this case, when the callback procedure completes and the SchedulePack package saves the record in the events table, the value of this variable will be assigned to the record-ID field. It is within (or just before) the Action-Code “mnemonic” SXkNewAfter is in effect that the SX\_PKey variable should be populated so the Events table ID field will be assigned its value. It is the 4D developer's responsibility to make sure that the value assigned to this variable is indeed a unique key for the events table.

If no value is assigned to this variable by the callback procedure (or if this procedure isn't defined), the SchedulePack package will derive its own unique key value for the record using 4D's Sequence number function call.

To see an example, please refer to the [“Callback Examples” on page 95](#).

#### **SX\_StartDt**

This date variable will be used to convey the starting-date value for the given event, corresponding to the start-date field for the event's record in the database's events table.

#### **SX\_EndDt**

This date variable will be used to convey the ending-date value for the given event, corresponding to the end-date field for the event's record in the database's events table.

#### **SX\_StartTm**

This time variable will be used to convey the starting-time value for the given event, corresponding to the start-time field for the event's record in the database's events table.

To see an example, please refer to the [“Callback Examples” on page 95](#)

**SX\_EndTm**

This time variable will be used to convey the ending-time value for the given event, corresponding to the end-time field for the event's record in the database's events table.

To see an example, please refer to the [“Callback Examples” on page 95](#).

**SX\_Summary**

This String variable will be used to convey the value of the event label for the given event, corresponding to the end-time field for the event's record in the database's events table. This is the string value that is entered by the user when the event is created, and is displayed in the event box on the grid.

**SX\_ColorIdx**

This integer variable will be used to convey the value of the color-palette index for the color used to fill the event box when it is drawn on the grid. It corresponds to the color-index field for the event's record in the database's events table.

When an event is created on the grid by a user, it is assigned a default color-index value (blue), but the application's new-event callback procedure can give the event a new color by assigning a different index value to this variable.

Likewise, the application can allow the user to assign a new color to the event, by providing a dialog with the appropriate fields/controls to be displayed by the double-click-event callback procedure.

To see an example, please refer to the [“Callback Examples” on page 95](#).

**SX\_CTagFKey**

Formerly SX\_ColumnID, this long integer variable will be used by the program to convey the value of the ID number of the column resource associated with the given event, corresponding to the column-ID field for the event's record in the database's events table.

To see an example, please refer to the [“Callback Examples” on page 95](#).

**SX\_ETagFKey**

Formerly SX\_LabelID, this long integer variable will be used by the program to convey the value of the ID number of the label resource associated with the given event, corresponding to the label-ID field for the event's record in the database's events table.

**SX\_Action**

This variable will be used to convey a code which identifies the specific user action that invoked the callback procedure. There is a range of possible values for actions which are discussed in the following paragraphs.

To see an example, please refer to the [“Callback Examples” on page 95](#).

### **SX\_EvtType**

This integer variable will be used to convey whether the event is in the Banner (1), or a Timed event (0).

### **SX\_Param1**

This is a generally used long integer variable which is currently only used by the SXkArraysBefore and SXkArraysAfter actions.

### **SX\_Picture**

This Picture variable will be used to convey the picture value for the given event, corresponding to the Picture field for the event's record in the database's events table.

### **SX\_Color**

This long integer variable will be used to convey the 24-bit color value for the color used to fill the event box when it is drawn on the grid. It corresponds to the 24-bit color field for the event's record in the database's events table.

### **SX\_TextColor**

This long integer variable will be used to convey the 24-bit color value for the color of the event's text. It corresponds to the 24-bit color text field for the event's record in the database's events table.

### **SX\_APopupMenuItems**

This two-dimensional text array variable can be filled in at the time of a 'SXkContextualBefore' action. The elements from this array will be displayed on the contextual menu.

### **SX\_PopupDevRow**

This long integer variable will contain the row value chosen by the User on a 'SXkContextualAfter' action. The row corresponds to a row in the 'SX\_APopupMenuItems' array.

### **SX\_PopupDevCol**

This long integer variable will contain the column value chosen by the User on a 'SXkContextualAfter' action. The column corresponds to a column in the 'SX\_APopupMenuItems' array.

### **SX\_TipText**

This Text variable should be assigned the desired text that is to be displayed as a tool tip on a 'SXkOnTip' action.

The following four variables provide information regarding the background area of a given cell area. Use the other SchedulePack variables provided to obtain information specific to an event record clicked.

**SX\_ClickResourceID**

This longint variable conveys the resource ID value for the given cell block clicked.

**SX\_ClickDt**

This date variable conveys the date value for the given cell block clicked.

**SX\_ClickStartTime**

This time variable conveys the beginning time value for the given cell block clicked.

**SX\_ClickEndTime**

This date variable conveys the ending time value for the given cell block clicked.

The results of clicking on these areas are shown in the following table:

	<b>SX_ClickResourceID</b>	<b>SX_ClickDt</b>	<b>SX_ClickStartTime</b>	<b>SX_ClickEndTime</b>
<b>Resource View</b>	Record ID from Resource column	Current Date	Cell Start Time	Cell End Time
<b>Month View</b>	Ignore	Date from cell clicked	Ignore	Ignore
<b>Date View</b>	Ignore	Date from column clicked	Cell Start Time	Cell End Time
<b>Timeline</b>	Record ID from Resource row	Current Date	Cell Start Time	Cell End Time

## Background Click Detection

### Action Codes

The routines in this section simply return a constant integer value which can be used as a comparison reference for the **SX\_Action** variable in the 4D callback procedure.

The callback mechanism uses a set of pre-defined 4D global variables to enable the SchedulePack package and the callback procedure to communicate with each other. One of these variables, named **SX\_Action**, will contain a special coded value that will enable the callback procedure to determine the user event or action that caused the callback procedure to be invoked.

This action-code value is used in a manner much like the concept of execution “phases”, used by 4D to enable layout procedures to determine the state of the layout and the task they are to perform each time they are called.

Because the callback procedure is assigned to the SchedulePack area procedurally, rather than being permanently “attached” to the object as a 4D object script, it is easy for the 4D developer to write simpler and more efficient procedures, tailored to specific classes of users or other application-defined situations, and assign them to the area as needed.

For each possible action-code value, SchedulePack provides a routine which can be used as a mnemonic reference to the value in the callback procedure instead of the hard-coded value. For example, if a callback procedure needs to test the **SX\_Action** variable to determine when the user have begun to draw a new event box on the grid, instead of coding the action-code value as a numeric constant as shown below:

### Incoming Action Codes

---

#### SXkNewArea

**Description:**

A SchedulePack area object has been initialized and is just about to be displayed on a new layout. This routine returns the reference value of the **SX\_Action** callback variable.

**Result:**

**SX\_Action** is set to -1.

**Syntax:**

`:(SX_Action = SXkNewArea) `New SchedulePack area being initialized`

---

#### SXkNewBefore

**Description:**

The user has begun a drag on the grid which will define a new event. Because the new event has not yet been created, the values of the other callback variables will not be defined - the only information the callback procedure can make use of is the value of **SX\_Action**.

When this callback occurs, the callback procedure can perform any validation necessary to allow/prevent new events from being created. For example, if the application only wants specific users to be allowed to create new events, the callback procedure would enforce this restriction at this time. If the callback procedure returns false, the user's drag will be ignored - the marquee will not be drawn and no new event will be created.

**Result:**

**SX\_Action** is set to 1.

**Syntax:**

`:(SX_Action = SXkNewBefore) `User drag to create event detected`

---

#### SXkNewAfter

**Description:**

The user has completed a drag on the grid which will define a new event.

The values of the other **SX\_** variables will be set to reflect the times and dates for the new event. The callback procedure can use these values for validation, etc., and can modify them before returning. For

example, if the application wants to assign a default label to new events, the callback procedure would do so at this time, by assigning a Long Integer to the **SX\_ETagFKey** variable.

One important exception is the **SX\_PKey** variable - since the new event has not yet been saved to the database, this variable will equal zero. If there are application-defined fields in the area's Events table that need to be initialized, the callback procedure must wait until it gets called with the **SX\_Action** variable equal to **SXkNewEdit** (3). That callback will occur immediately following this one if a value of true is returned.

When this callback occurs, the callback procedure can validate the new event. For example, if the application wants to confirm the creation of events during non-business hours, it would trap for this action-code value and then analyze the values of the **SX\_StartTm** and **SX\_EndTm** callback variables; if these values are outside the appropriate range, the procedure could be written to either ask the user to confirm them, to change the values to the nearest valid time frame, or to reject the new event by returning a false result value.

If the callback procedure returns true, the event will be saved as a new record in the area's Events table in the database. If the callback procedure returns false, the user's drag will be ignored - no new event will be created.

**Result:**

**SX\_Action** is set to 2.

**Syntax:**

:(**SX\_Action** = **SXkNewAfter**) `User drag to create event completed

---

### **SXkNewEdit**

**Description:**

The user has completed a drag on the grid to create a new event, and that event has just been saved to the database.

The values of the callback variables will be set to reflect the characteristics of the new event, including the **SX\_RecordNo** variable, whose value can be used to search for the event's record in the area's Events table.

When this callback occurs, the callback procedure can initialize any application defined fields in the area's Events table.

If the callback procedure returns true, the event box will become selected, and the user will be able to edit its text label. If the returned value is false, the event box will not be selected.

**Result:**

**SX\_Action** is set to 3.

**Syntax:**

:(**SX\_Action** = **SXkNewEdit**) `New event stored in database, ready for user edits

---

### **SXkMoveBefore**

**Description:**

The user has begun a drag on an event box, which will change the event's location on the scheduling grid. This action will also change the event's start/end times, start/end dates and/or resource ID.

If the callback procedure returns false, the user's drag will be ignored - the marquee will not be drawn and the event will retain its current parameters.

**Result:**

**SX\_Action** is set to 10.

**Syntax:**

:(**SX\_Action** = **SXkMoveBefore**) `User drag to move an event detected

---

### SXkMoveAfter

**Description:**

The user has completed a drag on an event box, which will change the location on the scheduling grid. If the callback procedure returns false, the user's drag will be ignored - the event will retain its previous parameters.

**Result:**

**SX\_Action** is set to 11.

**Syntax:**

:(**SX\_Action** = **SXkMoveAfter**) `User drag to move an event completed

---

### SXkResizeBefore

**Description:**

This routine returns the reference value of the **SX\_Action** callback variable when the user has begun a drag on the bottom border of an event box, which will change the event's end time (20). If the callback procedure returns false, the user's drag will be ignored - the marquee will not be drawn and the event will retain its current end time.

**Result:**

**SX\_Action** is set to 20.

**Syntax:**

:(**SX\_Action** = **SXkResizeBefore**) `User drag to resize an event detected

---

### SXkResizeAfter

**Description:**

This routine returns the reference value of the **SX\_Action** callback variable when the user has completed a drag on the bottom border of an event box, which will change the event's end time. If the callback procedure returns false, the user's drag will be ignored - the event will retain its previous end time.

---

**Result:**

`SX_Action` is set to 21.

**Syntax:**

`:(SX_Action = SXkResizeAfter) `User drag to resize an event completed`

---

### SXkEditBefore

**Description:**

This routine returns the reference value of the **SX\_Action** callback variable when the user has clicked on an event box or pressed Command-Tab, in an effort to “select” the event and edit its text box. If the callback procedure returns false, the user will not be able to edit the text label - the event will be “selected”, but will not be editable.

**Result:**

`SX_Action` is set to 30.

**Syntax:**

`:(SX_Action = SXkEditBefore) `User click to edit text of an event detected`

---

### SXkEditAfter

**Description:**

This routine returns the reference value of the **SX\_Action** callback variable when the user has “deselected” an event box that was previously selected, and may have modified the event's text box. If the callback procedure returns false, the user's edits to the label will be discarded.

**Result:**

`SX_Action` is set to 31.

**Syntax:**

`:(SX_Action = SXkEditAfter) `User click to edit text of an event completed`

---

### SXkColorBefore

**Description:**

This routine returns the reference value of the **SX\_Action** callback variable when the user has pressed the mouse button on an event box while holding down the control key, in order to invoke SchedulePack's color popup menu (40). If the callback procedure returns false, the mouse-click event will be ignored, and the color popup menu will not be displayed.

**Result:**

**SX\_Action** is set to 40.

**Syntax:**

`:(SX_Action = SXkColorBefore) `User control-click to edit color of an event detected`

---

### SXkColorAfter

**Description:**

This routine returns the reference value of the **SX\_Action** callback variable when the user has invoked SchedulePack's color popup menu and selected a color.

If the callback procedure returns false, the selected color will be ignored, and the event will be drawn using its original color.

**Result:**

**SX\_Action** is set to 41.

**Syntax:**

`:(SX_Action = SXkColorAfter) `User control-click to edit color of an event completed`

---

### SXkDelete

**Description:**

This routine returns the reference value of the **SX\_Action** callback variable when the user has pressed the forward-delete key while an existing event box was selected, in an effort to delete the event. From the runtime environment, a User can delete an event by simply clicking an event object and depressing the "forward" delete key. Alternatively, the developer can provide a scripted button which checks the **SX\_PKey** callback variable and deletes the corresponding event record.

If the callback procedure returns false, the keyboard event will be ignored, and the event will not be deleted.

**Result:**

**SX\_Action** is set to 51.

**Syntax:**

`:(SX_Action = SXkDelete) `User forward delete key to delete an event detected`

---

### SXkDoubleClick

**Description:**

This routine returns the reference value of the **SX\_Action** callback variable when the user has double-clicked on an event box. SchedulePack normally takes no special action when an event box is double-clicked, so the callback procedure's return value is irrelevant. This callback event allows the application to add special meaning to double-click events.

If the callback procedure returns true, the event box will become selected, and the user will be able to

edit its text label. If the returned value is false, the event box will not be selected.

The values below are used by the 4D callback procedure to request an action from the SchedulePack package. The callback procedure can request any one of these actions from the package, by assigning the corresponding value to the **SX\_Action** variable before it completes. The requested action will be taken regardless of the result value (true or false) returned by the callback procedure.

**Result:**

**SX\_Action** is set to 52.

**Syntax:**

`:(SX_Action = SXkDoubleClick) `User double-check on an event detected`

---

### SXkQueryBefore

**Description:**

This Callback event is new under version 3 and allows the 4D developer to perform their own query and to bypass SchedulePack's automatic query. This action comes immediately before "**SXkArraysBefore**". The callback should return "False" (\$0:=False) to allow the 4D Developer to perform their own query. Otherwise, the callback should return "True" (\$0:=True) if the 4D Developer wants SchedulePack to perform an automatic query.

There are several advantages to querying directly. For very large record selections, this will provide the 4D Developer the necessary controls to limit the record selection to a more manageable number of records. Within a SchedulePack display area, it may be ideal to show specific activity record subsets. Used in conjunction with the **SXkArraysBefore** and the **SXkArraysAfter** callback events, the 4D programmer has greater control over what is shown to the end-user.

If the Callback receives an "**SXkQueryBefore**" Action code (122), the **SX\_Param1** variable will contain a value of (1,2, or 3). This signifies which table will be queried, as follows:

- 1 - Events Table
- 2 - Labels Table
- 3 - Columns Table

**Result:**

**SX\_Action** is set to 122.

**Syntax:**

`:(SX_Action = SXkQueryBefore) ` Query 4D records directly.`

---

### SXkArraysBefore

**Description:**

Global arrays are temporarily used by SchedulePack to create Events. At the time of this action, the records have been selected. The 4D programmer may change the selection. If the callback returns TRUE, the arrays will be filled with the selected records. An "**SXkArraysAfter**" action will then occur before the Events are created from the arrays.

"**SX\_APKey**" array contains the Record IDs of the corresponding Records in the Database. Values less than or equal to zero need not have a Record in the Database.

The arrays are listed below:

"SX\_APKey" - LONGINT

Record ID

"SX\_AEvtText" - STRING[80]

Summary

"SX\_AStartDt" - DATE

Start Date

"SX\_AEndDt" - DATE

End Date

"SX\_AStartTm" - LONGINT

Start Time

"SX\_AEndTm" - LONGINT

End Time

"SX\_AClrIdx" - INTEGER

Color Index

"SX\_AEvtType" - INTEGER

Event Type

"SX\_AETgFKey" - LONGINT

Record ID of Label

"SX\_ACTgFKey" - LONGINT

Record ID of Column

At the time of this action another global variable, "**SX\_Param1**", will contain a value of (1,2, or 3) which signifies which table will be loaded into the arrays.

1 - Events Table ( all 10 arrays used )

2 - Labels Table ("SX\_APKey", "SX\_AEvtText" and "SX\_AClrIdx" only used )

3 - Columns Table ("SX\_APKey" and "SX\_AEvtText" only used )

**Result:**

**SX\_Action** is set to 120.

**Syntax:**

:(**SX\_Action = SXkArraysBefore**) `Selected records are about to be loaded into arrays

### SXkArraysAfter

**Description:**

This action comes immediately after "**SXkArraysBefore**". This action is exactly the same as "**SXkArraysBefore**" except the arrays have been loaded from the selected records. The arrays may be modified as necessary at this time. If the callback returns TRUE, the Events will then be created from the arrays.

**Result:**

**SX\_Action** is set to 121.

**Syntax:**

`:(SX_Action = SXkArraysAfter) `Arrays have been filled with selected records`

---

### SXkContextualBefore

**Description:**

This routine returns the reference value of the **SX\_Action** callback variable before a contextual menu is displayed. The '**SX\_PKey**' variable will contain the record ID of the event that was clicked on (a value of zero if an event was not clicked on). The developer has the option of filling in a two-dimensional array named '**SX\_APopupMenuItems**'. Elements from the array will be appended to the contextual menu.

**Result:**

**SX\_Action** is set to 124.

**Syntax:**

`SX_Action:= SXkContextualBefore `Similar to an on load phase for a contextual menu.`

---

### SXkContextualAfter

**Description:**

This routine returns the reference value of the **SX\_Action** callback variable after a contextual menu has been displayed. The '**SX\_PopupDevRow**' & '**SX\_PopupDevCol**' variables will contain the row & column chosen of the '**SX\_APopupMenuItems**' array.

**Result:**

**SX\_Action** is set to 125.

**Syntax:**

`SX_Action:= SXkContextualAfter `provides status after a contextual menu has been dismissed.`

### SXkOnDrop

**Description:**

This routine returns the reference value of the **SX\_Action** callback variable when something has been dropped onto the SchedulePack area. A call to '**SXGetDroppedObject**' will provide the drop information.

**Result:**

**SX\_Action** is set to 126.

**Syntax:**

**SX\_Action:= SXkOnDrop** `provides status after an object is dropped.

---

### SXkDragCopyBefore

**Description:**

This routine returns the reference value of the **SX\_Action** callback variable before an event is to be drag copied.

**Result:**

**SX\_Action** is set to 127.

**Syntax:**

**SX\_Action:= SXkDragCopyBefore** `before a drag copy

---

### SXkDragCopyAfter

**Description:**

This routine returns the reference value of the **SX\_Action** callback variable after an event has been drag copied.

**Result:**

**SX\_Action** is set to 128.

**Syntax:**

**SX\_Action:= SXkDragCopyAfter** `after a drag copy

---

### SXkOnTip

**Description:**

This routine returns the reference value of the **SX\_Action** callback variable when the mouse is hovering over an event. The text assigned to the '**SX\_TipText**' variable will be displayed as a tool tip. Assign an empty string if no tool tip is desired. The 'UseToolTips' parameter must be set to 1 in an earlier call to '**SXEventOpt**' for this action to happen.

**Result:**

**SX\_Action** is set to 129.

**Syntax:**

**SX\_Action:= SXkOnTip** `mouse is over an event

## Outgoing Action Codes

---

### SXkDoReload

**Description:**

This routine returns the value to be assigned to the **SX\_Action** callback variable when the application needs to request that SchedulePack reload the current event record from the database after the callback procedure has completed. This command is typically set within the SchedulePack callback procedure after the 4D developer has implemented adds or changes of an event record.

**Result:**

**SX\_Action** is set to 101.

**Syntax:**

**SX\_Action:= SXkDoReload** `Package reloads the event record from the database

---

### SXkDoRefresh

**Description:**

This routine returns the value to be assigned to the **SX\_Action** callback variable when the application needs to request that SchedulePack reload the entire event list from the database after the callback procedure has completed. The entire events-list is redrawn on the grid. This command is typically set within the SchedulePack callback procedure after the 4D developer has implemented adds or changes of one or more event records which fall within the current SchedulePack date or resource display range.

**Result:**

**SX\_Action** is set to 102.

**Syntax:**

**SX\_Action:= SXkDoRefresh** `Package reloads & redraws all Events records

## User Tips

### Creating Events/Objects

To create an Event record, the user can position the mouse pointer within the SchedulePack area and then point, click and drag a rectangle within the date (or resource) and time area desired.

If you wish to create an event record/object (physically) on top of an existing Event/object, you must

depress and hold the option key down and then perform the point-click & drag operation.

### Deleting Events/Objects

To delete an Event record, the user can select an event object within the SchedulePack area, then press Control-Delete (Windows) or Command-Delete (Macintosh).

### Moving Events/Objects

To move an Event record, the user can point, select and drag an object to another cell within the SchedulePack area. Dragging between non-timed Banner events and regular events and vice versa is not permitted.

### Resizing Events/Objects

To resize an Event record, the user must position the mouse pointer at the bottom of the Event object. Once the cursor changes to the cross-heir status, the use can click on the bottom of the object and drag it downward to establish a new end time. Whether creating or moving an object, it is only possible to create an end-time which corresponds to the currently shown interval. As a developer, either provide an interval tool which allows end-user to alter the time intervals shown or provide direct access to an input dialog or form for editing.

### Coloring Events/Objects

To change the color of an Event/object, the user can control-click on an event/object which then automatically displays the Color palette pop-up. The User can then drag the mouse pointer to the desired color cell and release the mouse to select that color. To cancel the color pop-up, the User can simply drag off of the palette.

### Tabbing

SchedulePack manages and maintains a list of selected cells throughout an application session. By depressing the Tab key the User can Tab between the events most recently edited. Alternatively, by depressing the Shift and Tab keys concurrently, the User can tab among these events in reverse order.

### Calendar Navigation

When the area is displayed to the user, they may select any date in the currently displayed month simply by clicking on it. The calendar will always have exactly one date selected. The user may scroll to the next or previous month by clicking on the arrow buttons in the upper-right or upper-left corner of the area. If the user holds down the Command key when clicking on these buttons, the calendar will scroll to the next or previous year.

Scrolling the calendar does not change the selected date - the user must click on a date cell in order to do so. If the currently selected date is in a month other than the one currently displayed in the area, the user may "jump" to the month of the currently selected date by clicking on the title-bar

## Command Summary:

### Activation

**SXRegister**(LicenseName(S);LicenseCodeMac(S);LicenseCodeWin(S))->String(S)

**SXShowDates**(xArea(L);StartDate(D);EndDate(D);ResourceIDs(array L))->L

**SXShowResources**(xArea(L);Date(D);ResourceIDs(array L))->L

## Preferences

**SXRowHdrFont**(xArea(L);FontName(S);FontSize(I);FontStyle(I);RowHeight(I);FontColor(L))

**SXColHdrFont**(xArea(L);Part(I);24bitColor(L))

**SXSetColor**(xArea(L);FontName(S);FontSize(I);FontStyle(I);FontColor(L))

**SXEventFont**(xArea(L);FontName(S);FontSize(I);FontStyle(I);FontColor(I))

**SXEvtLabelFont**(xArea(L);FontName(S);FontSize(I);FontStyle(I);FontColor(I))

**SXMinColWidth**(xArea(L);ColWidth(I))

**SXDateFormat**(xArea(L);FormatCode(I))

**SXTimeFormat**(xArea(L);TimeCycle(I);SuppressSuffix(I))

**SXAreaOpt**(xArea(L);SetHorixScroll(I);SetVertScroll(I);RealEventTimeTracking(I))

**SXEventOpt**(xArea(L);CanMakeBnr(I);CanColorBnr(I);CanMakeEvt(I);CanColorEvt(I);OverlapIn  
dent(I);UseGradiantFills(I);UseToolTips(I))

**SXDragCreateOpt**(xArea(L);CanMakewideBnr(I);CanMakewideEvt(I);CanMakeTallEvt(I))

**SXDragMoveOpt**(xArea(L);CanDragBnr(I);CanDragEvtHoriz(I);CanDragEvtVert(I))

**SXDragResizeOpt**(xArea(L);CanResizeBnr(I);CanResizewideEvt(I);CanResizeTallEvt(I))

**SXCreateStyleSheet**(xArea(L);ID(L);FontName(S);FontSize(I);FontStyle(I);FontColor(L)) ->L

**SXApplyStyleSheet**(xArea(L);ID(L);Part(I)) ->L

**SXDeleteStyleSheet**(xArea(L);ID(L)) ->L

## Banners

**SXBannerRow**(xArea(L);RowHeight(I))

## Configuration

**SXTimeIntervals**(xArea(L);Start(H);End(H);BHStart(H);BHEnd(H);IntervalMode(I))->L

**SXCallbackProc**(xArea(L);CallbackProc(S))

**SXEventsTable**(xArea(L);EventsTableNo(I);RecIDFld(I);StartDtFld(I);EndDtFld(I);StartTmFld(I);  
EndTmFld(I);SummaryFld(I);ColorFld(I);EventType(I);TextColorFld(I);IconFld(I))->L

**SXLabelsTable**(xArea(L);EventLblIDFld(I);LblFileNo(I);LblFileIDFld(I);LblFileStrFld(I);  
LblColorFld(I);LblTextColorFld(I))->L

**SXColumnsTable**(xArea(L);EventColIDFld(I);ColFileNo(I);ColFileIDFld(I);ColFileStrFld(I))->L

**SXGetScroll**(xArea(L);Row(I);Column(I))

**SXSetScroll**(xArea(L);Row(I);Column(I))->L

**SXRefresh**(xArea(L))->L

**SXGetView**(xArea(L);ViewKind(I)) ->L

**SXSetView**(xArea(L);ViewKind(I)) ->L

### Utilities

**SXGetSelected**(xArea(L);RecordID(L))->L

**SXDeleteEvent**(xArea(L);RecordID(L))->L

**SXColorPopup**(CurrentColor(I);ColorNum(I))

**SXRowToTime**(xArea(L);Row(I);Time(H))->L

**SXTimeToRow**(xArea(L);Time(H);Row(I))->L

**SXColorPicker**(CurrentColor(L);PickedColor(L))->L

**SXColorToIndex**(24BitColor (L))->I

**SXColorToRGB**(24BitColor(L);RedComponent(I);GreenComponent(I);BlueComponent(I))

**SXExport**(xArea(L);TimeZoneKind(I);TimeZoneValue(S);DestFilePath(T))->L

**SXGetDraggedEvent**(xArea(L))->L

**SXGetDroppedObject**(xArea(L);SourcePtr(Z);SourceElement(L); SourceProcess(L); DestDate(D); DestResourceID(L); DestStartTime(H); DestEndTime(H))->L

**SXImport**(xArea(L);MinutesOffset(L);SourceFilePath(T))->L

**SXIndexToColor**(ColorIndex(I))->L

**SXRGBToColor**(RedComponent(I); GreenComponent(I); BlueComponent(I))->L

### Calendar Control

**SXCalFormat**(xArea(L);Dayformat(I);Monthformat(I))

**SXCalFonts**(xArea(L);HeaderFont(S);HeaderFontSize(I);HeaderFontStyle(I);CellFont(S); CellFontSize(I);CellFontStyle(I))

**SXCalDateColor** (xArea(L);DayNumber(I);TextColor(I))->L

**SXCalDateIcon**(xArea(L);DayNumber(I);IconLocation(I);IconIndex(I);IconColor(I))->L

**SXCalClear** (xArea(L))->L

**SXCalGetMonth** (xArea(L);CurrentMonth(I))->L

**SXCalGetYear**(xArea(L);CurrentYear(L))->L

**SXCalSetMonthYr** (xArea(L);Month(I);Year(L))

**SXCalGetDate**(xArea(L);SelectedDate(D))->L

**SXCalSetDate** (xArea(L);DateToSelect(D))

## Month View Control

**SXMonthClearPictures**(xArea(L);FromDate(D);ToDate(D)) ->L

**SXMonthFont**(xArea(L);Part(I);Font Name(S);Font Size(I);Font Style(I);Font Color(I)) ->L

**SXMonthGetMonthYear**(xArea(L);Month(I);Year(I)) ->L

**SXMonthOpt**(xArea(L); StartDayOfWeek Start day of week(I); ShowNonMonthDays Show non-month days(I);EventOrder(I)) ->L

**SXMonthSetColor** (xArea(L);MonthPart(I);YearColor(IL)) ->L

**SXMonthSetDatePicture**(xArea(L);Date(D);Picture(P);HorizLoc(I);VerticalLoc(I);SizeType(I);MaxSize(I)) ->L

**SXMonthSetDayColor** (xArea(L);DayNumber(I);Color(LI)) ->L

**SXMonthSetMonthYear** (xArea(L);Month(I);Year(I)) ->L

## Timeline View Control

**SXTimeLineFont**(xArea(L);Part(I);Fontname(S); Fontsize(I); Fontstyle(I);FontColor(L))->L

**SXTimeLineOpt**(xArea(L);ColumnWidth(I);ColumnHeaderHeight(I);ResourceHeaderWidth(I);ResourceHeight(I);ColumnGridLinesSize(I);ResourceGridLinesSize(I);TimeGridLinesSize(I))->L

**SXTimeLineSetColor** (xArea(L);Part(I);Color(L))->L

## Chart Control

**SXChartColumns**(Area:(L);Alignment:(I);Headers:&X;Start time value:(L);End time value:(L))->L

**SXChartResources**(Area:(L);Header:(T);Resources:(Array of &X))->L

**SXChartSetResourceValues**(Area:(L);Resource Index:(L);Start time values:(Array L);End time values:(Array L);Colors:(Array I)) ->L

**SXChartColHdrOpt**(xArea:(L);Column Width:(I);Header height:(I);Font name:(S);Font size:(I);Font style:(I);Font color:(I);Background color:(I))->L

**SXChartResourcesOpt**(xArea(L);Header width:(I);Row height:(I);Font name:(S);Font size:(I);Font style:(I);Font color:(I);Background color:(I))->L

**SXChartGridOpt**(Area:(L);Horizontal lines visible:(I);Horizontal lines size:(I);Horizontal lines color:(I);Vertical lines visible:(I);Vertical lines size:(I);Vertical lines color:(I);Background color:(I))->L

**SXChartAreaOpt**(xArea:(L);Horizontal ScrollBar:(I);Vertical ScrollBar:(I))->L

## CallBack Support Variables

**SX\_PKey**

**SX\_StartDt**

**SX\_EndDt**

**SX\_StartTm**

**SX\_EndTm**

**SX\_Summary**

**SX\_ColorIdx**

**SX\_CTagFKey**

**SX\_ETagFKey**

**SX\_Action**

**SX\_EvtType**

**SX\_Param1**

**SX\_Picture**

**SX\_Color**

**SX\_TextColor**

**SX\_PopupDevRow**

**SX\_PopupDevCol**

**SX\_TipText**

**SX\_APopupMenuItems**

## Action Codes

Action Mnemonic	Equivalent Integer Value
<b>SXkNewArea</b>	-1
<b>SXkNewBefore</b>	1
<b>SXkNewAfter</b>	2
<b>SXkNewEdit</b>	3
<b>SXkMoveBefore</b>	10
<b>SXkMoveAfter</b>	11
<b>SXkResizeBefore</b>	20
<b>SXkResizeAfter</b>	21

<b>SXkEditBefore</b>	30
<b>SXkEditAfter</b>	31
<b>SXkColorBefore</b>	40
<b>SXkColorAfter</b>	41
<b>SXkDelete</b>	51
<b>SXkDoubleClick</b>	52
<b>SXkQueryBefore</b>	120
<b>SXkArraysAfter</b>	121
<b>SXkDoReload</b>	101
<b>SXkDoRefresh</b>	102
<b>SXkContextualBefore</b>	124
<b>SXkContextualAfter</b>	125
<b>SXkOnDrop</b>	126

## Error Codes

**-15001 - Invalid Event Table Number**

Solution: Change the Event table number to reflect a valid Event Table number.

**-15002 - Invalid Event Primary Key Field Type**

Solution: Change the Event Primary Key field to field type Long Integer.

**-15003 - Invalid Event Summary Field Type**

Solution: Change the Event Summary field to field type Alpha(80).

**-15004 - Invalid Start Date Field Type**

Solution: Change the Start Date field to field type Date.

**-15005 - Invalid End Date Field Type**

Solution: Change the End Date field to field type Date.

**-15006 - Invalid Start Time Field Type**

Solution: Change the Start Time field to field type Time.

**-15007 - Invalid End Time Field Type**

Solution: Change the End Time field to field type Time.

**-15008 - Invalid Color Field Type**

Solution: Change the Color field to field type Integer or Long Integer.

**-15009 - Invalid Event Type Field Type**

Solution: Change the Event Type Field to field type Integer.

**-15010 - Invalid Text Color Field Type**

Solution: Change the Text Color Field to field type Integer or Long Integer.

**-15011 - Invalid Icon Field Type**

Solution: Change the Icon Field to field type Picture.

**-15100 - Invalid Resource Table Foreign Key Field Type**

Solution: Change the Resource Table Foreign key to field type Long Integer.

**-15200 - Invalid Label Table Foreign Key Field Type**

Solution: Change the Resource Table Foreign key field type to Long Integer.

**-15101 - Invalid Column Table Number**

Solution: Change the Column Table number to correctly reflect the Column table.

**-15102 - Invalid Column Table Primary Key Field Type**

Solution: Change the Column Table Primary Key field type to Long Integer. This is used in a Resources view and used in conjunction with the SXSHow Resources command.

**-15103 - Invalid Column Table Display Field Type**

Solution: Change the Column Table display field type to Alpha (80). This is used in a Resources view and used in conjunction with the SXSHow Resources command.

**-15104 - Invalid Column Table Color Field Type**

Solution: Change the Column Table Color field type to Integer or Long Integer. This is used in a Resources view and used in conjunction with the SXSHow Resources command

**-15105 - Invalid Column Table Text Color Field Type**

Solution: Change the Column Table Text Color field type to Integer or Long Integer. This is used in a Resources view and used in conjunction with the SXSHow Resources command

**-15201 - Invalid Label Table Number**

Solution: Change and correct the Label Table number (this is the table for individual cell labels).

**-15202 - Invalid Label Table Primary Key Field Type**

Solution: Change and correct the Label Table number (this is table for cell labels).

**-15203 - Invalid Label Table Label Field Type**

Solution: Change and correct the Label Table field type to Alpha 80. Text is not a valid field type for the Label table Label field and should not be used.

**-15300 - Not a Long Integer Array**

Solution: Change the specified array type to Long Integer.

**-15301 - Not an Integer Array**

Solution: Change the specified array type to Integer.

**-15302 - Not a Text Array**

Solution: Change the specified array type to Text.

**-15400 - Invalid Date**

Solution: Change the specified field value to a legitimate date.

**-15401 - Invalid Time**

Solution: Change the specified field value to a legitimate time.

**-15402 - Invalid Time Interval Mode**

Solution: Change the time interval mode to a legitimate time interval.  
Acceptable values are 1, 2, 3, 4, 5, 6, 10, 12, 15, 20 and 30.

**-15403 - Invalid Month Day Number**

Solution: Change the day number for the current month to a valid day number.  
Acceptable values are determined by any legitimate calendar.

**-15410 - Invalid Font Name**

Solution: Use a font name that is available locally.

**-15411 - Invalid Font size**

Solution: Use a font size greater than zero.

**-15412 - Invalid Font style**

Solution: Use a font style greater than zero.

**-15413 - Invalid Font color**

Solution: Use a font color greater than or equal to zero.

**-15420 - Invalid StyleSheet ID**

Solution: Use a stylesheet ID that had been specified in a call to SXCreateStyleSheet.

**-15500 - Invalid Icon Index**

Solution: Change the icon index for the Small Resource icon to a valid index number for an existing resource.

**-15600 - Invalid Column Number**

Solution: Change the column number specified to a legitimate column number. For a "Date view", the number of days within the date range specified equals the number of columns. For a "Resource view", the size of the array specified within the SXShowResources command equals the number of legitimate

columns.

**-15601 - Invalid Row Number**

Solution: Change the row number specified to a legitimate row number for the current SchedulePack area. For either the Date or Resource view, the number of rows is equal to the number of hours specified within the SXTTimeInterval command times the number of intervals per hour. A number larger than this would generate the -15601 error code.

**-15602 - Attempt to Delete unknown record**

Solution: Correct the invalid reference to the record you want to delete. The record reference specified cannot be found.

**-15700 - Not a valid SchedulePack Area**

Solution: Ensure you are referencing a SchedulePack area.

**-15710 - Invalid TimeZone kind**

Solution: Correct the TimeZone specified

**-15711 - Invalid file path**

Solution: Correct the filepath name

**-15712 - Cannot open file for reading or writing**

Solution: Possible a permissions error. Correct the permissions for the current user for the file or file location in question.

# Programming Examples

The following structure is used for the method and form examples shown in this section. The Activities table, shown below, is where all of the “events” information is stored. Notice that there are 5 associated or related resources to the Activities table.

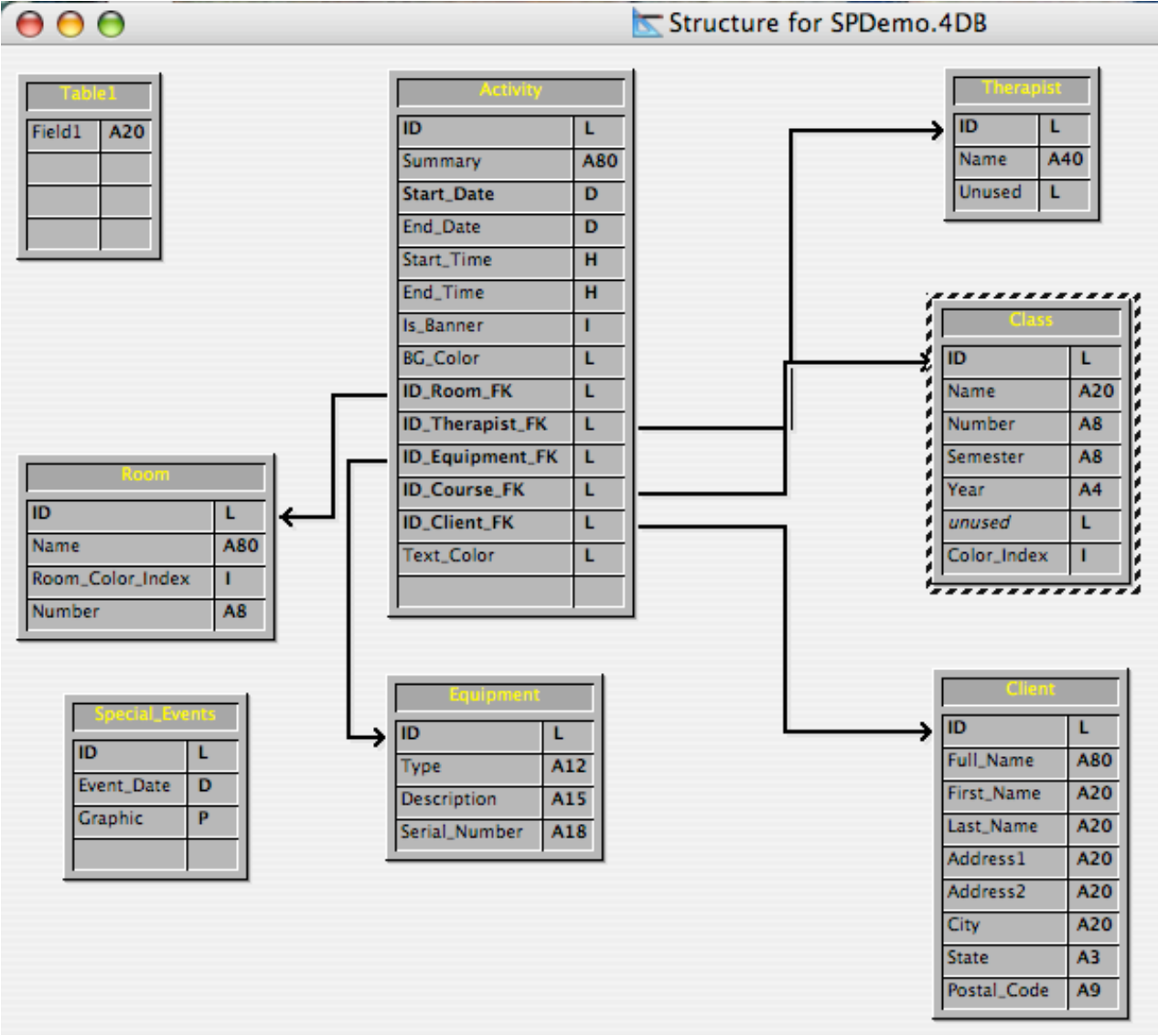


Figure 7. The Activities table

## SchedulePack Definition and Setup

Within your 4D application, you'll be required to define all forms or dialogs which include the SchedulePack area.

The Developer must name the SchedulePack area.

## Typical Command Sequence

Although there are over two dozen schedule related commands which provide a detailed level of control over the SchedulePack area, only a handful of commands are required to provide almost complete functionality. **Before using the SchedulePack commands, first declare (as in 4D compiler declarations) the SchedulePack variables.** These variables are found in the “Callback Variables” section of this manual. Failure to do so may yield unpredictable values and possibly crash the application, especially when working with source code. The essential SchedulePack commands and flow are as follows:

**SXEventsTable** is first called to define the events table to the **SchedulePack** area. The **SXCallbackProc** must be called prior to displaying the **SchedulePack** plug-in area. This method defines the 4D method which is executed or, alternatively, the developer can pass a null string to specify no callback method is enabled. The **SXBanners** command is called to indicate whether banner events (rows) are to be shown. The **SXLabelsTable** is called prior to the **SXShowResources** or **SXShow-Dates** commands in order to enable the Resource labeling feature and identify the resource table, the ID field and the label field this feature uses. The **SXColumnsTable** is called prior to calling the **SXShowResources** command to identify the table and fields for the resource group which comprises the desired column definitions for the grid. The **SXShowDates** command is called to configure a **SchedulePack** area to display events for a specific range of dates, with a column on the grid for each date. The **SXShowResources** command is called to display events on a single date for a specific group of resources, with a column on the grid for each resource.

**The SchedulePack commands shown above should initially be called where the first parameter is set to “0” (that’s zero). For more details, please see the “Command Reference:” section.**

## Application Definition and Setup

### **SXEventsTable**

‣tell SchedulePack what Table and Fields are to be used for Events.

‣ "SXEventsTable" parameters:

‣ ScheduleArea(L) - Schedule Area; 0 = default for New Schedule Areas

‣ EventsTableNum(I) - Table number of the Events Table

‣ RecordIDField(I) - Field number of the Record ID

‣ StartDateField(I) - Field number of the Start Date

‣ EndDateField(I) - Field number of the End Date

‣ StartTimeField(I) - Field number of the Start Time

‣ EndTimeField(I) - Field number of the End Time

‣ SummaryField(I) - Field number of the Summary

‣ ColorField(I) - Field number of the Color Index

‣ BannerFlagField(I) - Field number of the Banner Flag

```
$!_error:=SXEventsTable (0;Table (->[Activity]);Field(->[Activity]Act_PKey);Field(->[Activity]Act_SDate);Field(->[Activity]Act_EDate);
```

```
Field(->[Activity]Act_STime);Field(->[Activity] Act_ETime);
```

```
Field(>[Activity]Act_Summary);Field(>[Activity]Act_Summary);
```

```
Field(->[Activity]Act_Color_Index); Field(->[Activity]Act_Is_Banner))
```

```
If ($!_error # 0)
```

```

    Error_Alert ($!_error)
End if

```

#### SXEventsTable Error Codes

```

-- No Error = 0
-- InvalidXArea = -15700
-- BadEventTableNum = -15001
-- BadRecordIDFieldType = -15002
-- BadEventSummaryFieldType = -15003
-- BadStartDateFieldType = -15004
-- BadEndDateFieldType = -15005
-- BadStartTimeFieldType = -15006
-- BadEndTimeFieldType = -15007
-- BadColorIndexFieldType = -15008
-- BadEventTypeFieldType = -15009
-- InvalidTextColorFieldType = -15010
-- InvalidIconFieldType = -15011

```

### Declaration of Arrays for Related Resource Tables

To define which activity records which will display in a SchedulePack area, limit the arrays created for the related resource tables.

```

`Therapist_To_Array

ARRAY LONGINT (al_Thrpst_PKey;0)

ALL RECORDS ([Therapist])

SELECTION TO ARRAY([Therapist]Therapist_PKey;al_Thrpst_ID;[Therapist]
Therapist_Name;as_ThrpstTag)

```

### Callback Variable Declaration

DeclareVars Method; The 4D Developer may optionally declare the following compiler directives within your SchedulePack application. Please note these variables are already declared within the SchedulePack plug-in. Additionally, the SX\_Summary variable is internally declared as string(80) and will cause a compiler error if declared as something else within the 4D program.

```

C_INTEGER(SX_Action)

C_LONGINT(SX_PKey)

C_STRING(80;SX_Summary)

C_DATE(SX_StartDt;SX_EndDt)

C_TIME(SX_StartTm;SX_EndTm)

```

```
C_INTEGER(SX_ColorIdx)
C_INTEGER(SX_EvtType)
C_LONGINT(SX_ETagFKey)
C_LONGINT(SX_CTagFKey)
C_LONGINT(SX_Param1)
```

Currently, only the SXkArraysBefore and SXkArraysAfter Actions use these arrays, but they should be declared as follows:

```
ARRAY LONGINT(SX_APKey;0)
ARRAY LONGINT(SX_AStartTm;0)
ARRAY LONGINT(SX_AEndTm;0)
ARRAY LONGINT(SX_AETgFKey;0)
ARRAY LONGINT(SX_ACTgFKey;0)
ARRAY DATE(SX_AStartDt;0)
ARRAY DATE(SX_AEndDt;0)
ARRAY INTEGER(SX_AClrIdx;0)
ARRAY INTEGER(SX_AEvtType;0)
ARRAY STRING(80;SX_AEvtText;0)
```

### Action Variable Declaration

To be more efficient, variables should be used to hold Action values, such as SXkNewArea, SXkMoveBefore, SXkDoubleClick, and so on. These variables should be declared and initialized once, and then can be used in place of their corresponding Action functions.

```
`define Action variables
C_INTEGER(SX_kNewArea)
C_INTEGER(SX_kNewBefore)
C_INTEGER(SX_kNewAfter)
C_INTEGER(SX_kNewEdit)
C_INTEGER(SX_kEditBefore)
C_INTEGER(SX_kEditAfter)
C_INTEGER(SX_kMoveBefore)
C_INTEGER(SX_kMoveAfter)
C_INTEGER(SX_kResizeBefore)
```

**C\_INTEGER(SX\_kResizeAfter)**  
**C\_INTEGER(SX\_kColorBefore)**  
**C\_INTEGER(SX\_kColorAfter)**  
**C\_INTEGER(SX\_kDelete)**  
**C\_INTEGER(SX\_kDoubleClick)**  
**C\_INTEGER(SX\_kDoReload)**  
**C\_INTEGER(SX\_kDoRefresh)**  
**C\_INTEGER(SX\_kArraysBefore)**  
**C\_INTEGER(SX\_kArraysAfter)**

`retrieve Action values into variables

**SX\_kNewArea:=SXkNewArea**  
**SX\_kNewBefore:=SXkNewBefore**  
**SX\_kNewAfter:=SXkNewAfter**  
**SX\_kNewEdit:=SXkNewEdit**  
**SX\_kEditBefore:=SXkEditBefore**  
**SX\_kEditAfter:=SXkEditAfter**  
**SX\_kMoveBefore:=SXkMoveBefore**  
**SX\_kMoveAfter:=SXkMoveAfter**  
**SX\_kResizeBefore:=SXkResizeBefore**  
**SX\_kResizeAfter:=SXkResizeAfter**  
**SX\_kColorBefore:=SXkColorBefore**  
**SX\_kColorAfter:=SXkColorAfter**  
**SX\_kDelete:=SXkDelete**  
**SX\_kDoubleClick:=SXkDoubleClick**  
**SX\_kDoReload:=SXkDoReload**  
**SX\_kDoRefresh:=SXkDoRefresh**  
**SX\_kArraysBefore:=SXkArraysBefore**  
**SX\_kArraysAfter:=SxkArraysAfter**

### SchedulePack Area Views

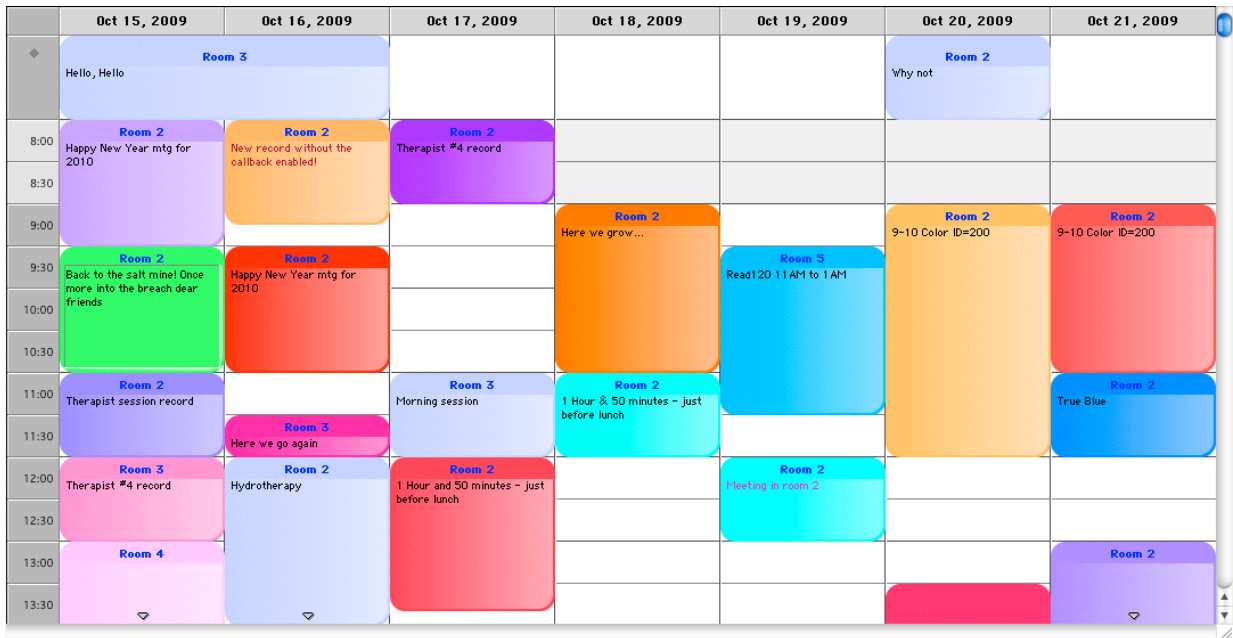


Figure 8. Date View

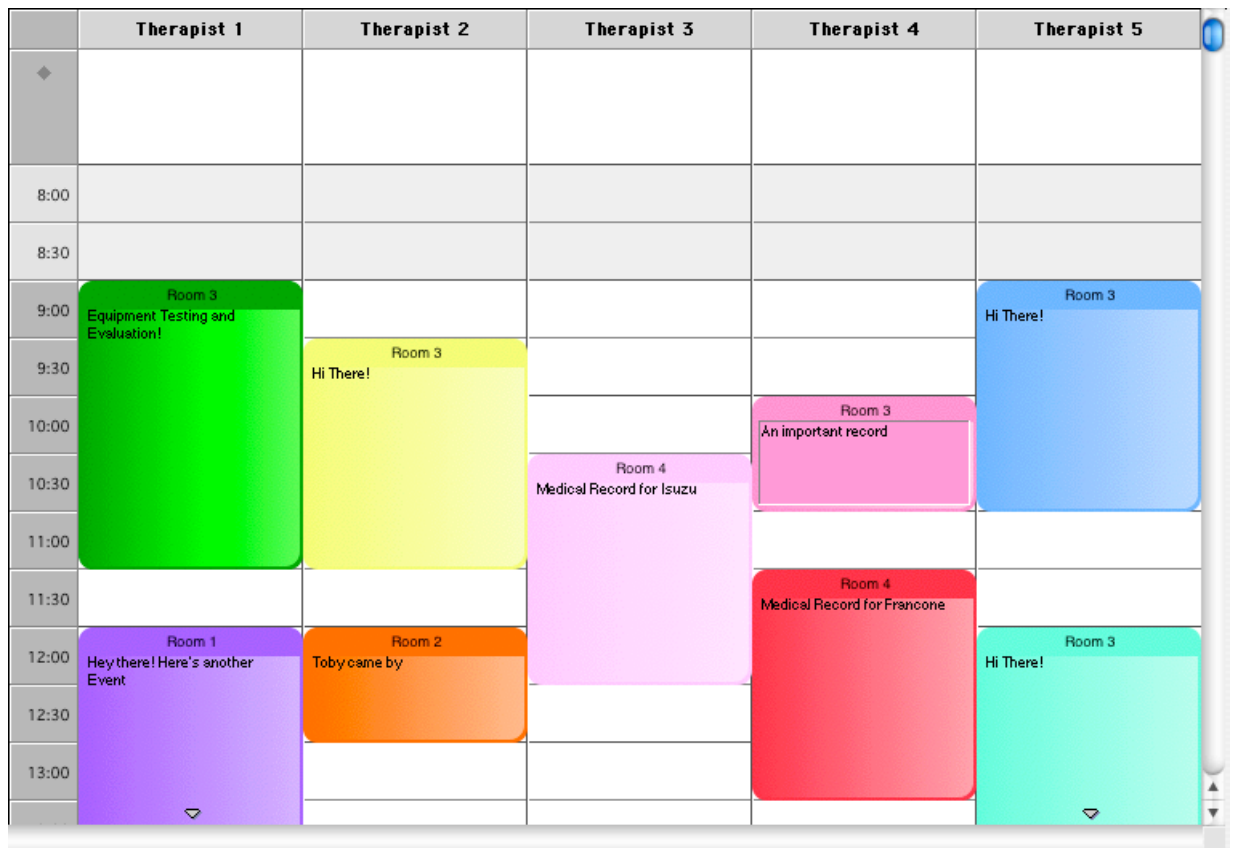


Figure 9. Resource View

`[Table1]SX

**C\_LONGINT**(\$l\_error;\$l\_formEvent)  
 \$l\_formEvent:=Form event

**Case of**

:( \$l\_formEvent=On Load )  
 b\_HorizontalScroll:=1 `Hide/Show Horizontal Scrollbar Checkbox  
 b\_VerticalScroll:=1 `Hide/Show Vertical Scrollbar Checkbox  
 b\_Banners:=1 `Hide/Show Banners Button  
 b\_CanCreateBanners:=1 `Create New Banner Events Checkbox  
 b\_CanCreateEvents:=1 `Create New Events Checkbox  
 b\_CanColorBanners:=1 `Enable/Disable Coloring of Banner Events Checkbox  
 b\_CanColorEvents:=1 `Enable/Disable Coloring of Events Checkbox  
 b\_RoundedEvents:=0 `Enable/Disable Rounded Events Checkbox  
 b\_ShowRoomLabelColors:=0

RulerIntervals:=6 `Ruler for number of days in Dates View

d\_endDate:=d\_beginDate+RulerIntervals

`Initialize 3 sets of 3x buttons

vToggleSet1a:=1  
 vToggleSet2a:=1  
 vToggleSet3a:=1  
 vToggleSet1b:=1  
 vToggleSet2b:=1  
 vToggleSet3b:=1  
 vToggleSet1c:=1  
 vToggleSet2c:=1  
 vToggleSet3c:=1

b\_ToggleOp1:=1  
 b\_ToggleOp2:=1  
 b\_ToggleOp3:=1

**BUTTON TEXT**(b\_ToggleOp1;"Wide Banner Creation") `Init button Labels  
**BUTTON TEXT**(b\_ToggleOp2;"Wide Event Creation")  
**BUTTON TEXT**(b\_ToggleOp3;"Tall Event Creation")

`View Choices

**ARRAY STRING**(10;as\_ViewChoice;6)  
 as\_ViewChoice{1}:= "Dates"  
 as\_ViewChoice{2}:= "Therapists"  
 as\_ViewChoice{3}:= "Classes"  
 as\_ViewChoice{4}:= "Rooms"  
 as\_ViewChoice{5}:= "Equipment"

```
as_ViewChoice{6}:="Clients"  
as_ViewChoice:=i_CurrentView
```

Time/Row Choices

```
ARRAY STRING(10;as_RowSize;6)
```

```
as_RowSize{1}:="1 Hour"  
as_RowSize{2}:="30 Minutes"  
as_RowSize{3}:="20 Minutes"  
as_RowSize{4}:="15 Minutes"  
as_RowSize{5}:="12 Minutes"  
as_RowSize{6}:="10 Minutes"  
as_RowSize:=2 `Default to "30 Minutes" choice
```

Hours Choices

```
ARRAY STRING(12;as_TimeInt;6)
```

```
as_TimeInt{1}:="9 AM - 5 PM"  
as_TimeInt{2}:="7 AM - 7 PM"  
as_TimeInt{3}:="6 AM - 9 PM"  
as_TimeInt{4}:="6 AM - 10 PM"  
as_TimeInt{5}:="6 AM - 11 PM"  
as_TimeInt{6}:="9 AM - 3 AM"  
as_TimeInt:=1
```

Option set Choices

```
ARRAY STRING(15;as_TogleSet;3)
```

```
as_TogleSet{1}:="Create Options"  
as_TogleSet{2}:="Resize Options"  
as_TogleSet{3}:="Drag Options"  
as_TogleSet:=1 `Default to "Create Options" choice  
as_TogleSet{0}:=as_TogleSet{1}
```

Increment/Decrement Choices

```
ARRAY STRING(4;as_Pop_Day_Week;2)
```

```
as_Pop_Day_Week{1}:="Day"  
as_Pop_Day_Week{2}:="Week"  
as_Pop_Day_Week:=1 `Default to "Day" choice  
as_Pop_Day_Week{0}:=as_Pop_Day_Week{1}
```

Variable to display the default Color

```
ARRAY STRING(10;as_Pop_Color;1)
```



**SXCalSetDate** (E\_CAL;d\_beginDate)

*SetCalDayMonthFormat* (as\_dateTimeFmt)

**SetDefaultColor** (Yellow )

**SXBannerRow** (E\_SCHED;-1) `Show Banners w/ default height

**SXDragCreateOpt** (E\_SCHED;1;1;1) `Enable all Creation capabilities

**SXDragResizeOpt** (E\_SCHED;1;1;1) `Enable all Resizing capabilities

**SXDragMoveOpt** (E\_SCHED;1;1;1) `Enable all Moving capabilities

**SXAreaOpt** (E\_SCHED;1;1;1) `Show Hor. & Vert Scroll Bar, +RT Tracking

` "SXEventOpt" parameters:

- ` ScheduleArea(L) - Schedule Area
- ` CanMakeBanner(I) - Can create New Banner Events; (-1 = don't change)
- ` CanColorBanner(I) - Can color Banner Events; (-1 = don't change)
- ` CanMakeEvent(I) - Can create New Events; (-1 = don't change)
- ` CanColorEvents(I) - Can color Events; (-1 = don't change)
- ` Indent(I) - Indent; (-1 = don't change)
- ` EventRoundAmount(I) - Event Round Amount; (-1 = don't change)
- ` GradientFills(I) - Gradient fills; (-1 = don't change)
- ` UseToolTips(I) - Use tool tips; (-1 = don't change)

**SXEventOpt** (E\_SCHED;1;1;1;14;75;1;0)

` "SXIndexToColor" parameters:

- ` ColorIndex(I) - Color index ( 0 to 255 )
- ` <-- return(L) 24-bit color

`\$whiteColor:=SXIndexToColor (White )

\$blackColor:=SXIndexToColor (Blue )

**SXEventFont** (E\_SCHED;"";-1;-1;\$blackColor) ` set event text to black

**SXEvtLabelFont** (E\_SCHED;"";-1;-1;\$blackColor) ` set event labels to black

**SetScheduleTimeFormat** (E\_SCHED;as\_dateTimeFmt)

**SetupRows** (E\_SCHED;as\_TimeInt;as\_RowSize)

**SwitchView** (i\_CurrentView)

**End case**

## Callback Examples

```

`SPack_Callback

C_BOOLEAN($0;$bo_AllowAction)
C_LONGINT($l_elemNum)
C_TEXT(t_oldTag;$t_newTag)
C_TIME(c_oldStartTime;c_oldEndTime)

```

```
$bo_AllowAction:=True
```

### Case of

```

: (SX_Action=SX_kNewArea)

: (SX_Action=SX_kQueryBefore) `Introduced in version 3

: (SX_Action=SX_kNewBefore) `

: (SX_Action=SX_kDoubleClick) `SX_kDoubleClick - Double-Clicked on an Event
  Edit_Event (SX_PKey) ` "SX_PKey" contains the Record ID of the Event
  SX_Action:=SX_kDoReload

: (SX_Action=SX_kNewEdit)
  `At this point the New Activity Record has been created and saved, but
  `probably is not in the record selection. "SX_PKey" contains Event Record ID
  Edit_Event (SX_PKey)
  SX_Action:=SX_kDoReload

: (SX_Action=SX_kNewAfter)
  `At this point the New Activity Record has NOT been created.
  `It will be created and saved after this action.
  ∅l_RecID:=∅l_RecID+1 `assign a unique ID
  SX_PKey:=∅l_RecID
  SX_ColorIdx:=i_DefaultColor

: (SX_Action=SX_kMoveBefore)
  If (i_CurrentView=1) `Dates View

      If (SX_PKey<0)
          $bo_AllowAction:=False `a Lunch Event; stop move
      End if

  Else
      `Save the Column Name to be used in the "Move After" action.
      ` "SX_CTagFKey" will contain the Record ID of the Column Record.
      t_oldTag:=""

```

**Case of****: (i\_CurrentView=2) `Therapists View****\$l\_elemNum:=Find in array(al\_Thrpst\_PKey;SX\_CTagFKey)****If (\$l\_elemNum>0)****t\_oldTag:=as\_ThrpstTag{\$l\_elemNum}****End if****: (i\_CurrentView=3) `Classes View****\$l\_elemNum:=Find in array(al\_Class\_PKey;SX\_CTagFKey)****If (\$l\_elemNum>0)****t\_oldTag:=as\_ClassNum{\$l\_elemNum}****End if****: (i\_CurrentView=4) `Rooms View****\$l\_elemNum:=Find in array(al\_Room\_PKey;SX\_CTagFKey)****If (\$l\_elemNum>0)****t\_oldTag:=as\_RoomTag{\$l\_elemNum}****End if****: (i\_CurrentView=5) `Equipment View****\$l\_elemNum:=Find in array(al\_Equip\_PKey;SX\_CTagFKey)****If (\$l\_elemNum>0)****t\_oldTag:=as\_EquipTag{\$l\_elemNum}****End if****: (i\_CurrentView=6) `Clients View****\$l\_elemNum:=Find in array(al\_Client\_PKey;SX\_CTagFKey)****If (\$l\_elemNum>0)****t\_oldTag:=as\_ClientTag{\$l\_elemNum}****End if****End case****End if `i\_CurrentView # 1****: (SX\_Action=SX\_kMoveAfter)****If (bPrompt=1) `prompt the User****If (i\_CurrentView=1) `Dates View****\$bo\_AllowAction:=Action\_Confirm ("Move Event to " +String(SX\_StartDt;5)+" at  
"+String(SX\_StartTm;5)+ "-" +String(SX\_EndTm;5)+"?")**

**Else**

  `"SX\_CTagFKKey" will contain the Record ID of the Column Record.  
  \$t\_newTag:= ""

**Case of**

  : (i\_CurrentView=2) `Therapists View

    \$l\_elemNum:=Find in array(al\_Thrpst\_PKey;SX\_CTagFKKey)

**If (\$l\_elemNum>0)**

      \$t\_newTag:=as\_ThrpstTag{\$l\_elemNum}

**End if**

  : (i\_CurrentView=3) `Classes View

    \$l\_elemNum:=Find in array(al\_Class\_PKey;SX\_CTagFKKey)

**If (\$l\_elemNum>0)**

      \$t\_newTag:=as\_ClassNum{\$l\_elemNum}

**End if**

  : (i\_CurrentView=4) `Rooms View

    \$l\_elemNum:=Find in array(al\_Room\_PKey;SX\_CTagFKKey)

**If (\$l\_elemNum>0)**

      \$t\_newTag:=as\_RoomTag{\$l\_elemNum}

**End if**

  : (i\_CurrentView=5) `Equipment View

    \$l\_elemNum:=Find in array(al\_Equip\_PKey;SX\_CTagFKKey)

**If (\$l\_elemNum>0)**

      \$t\_newTag:=as\_EquipTag{\$l\_elemNum}

**End if**

  : (i\_CurrentView=6) `Clients View

    \$l\_elemNum:=Find in array(al\_Client\_PKey;SX\_CTagFKKey)

**If (\$l\_elemNum>0)**

      \$t\_newTag:=as\_ClientTag{\$l\_elemNum}

**End if**

**End case**

**If (t\_oldTag=\$t\_newTag)**

  \$bo\_AllowAction:=Action\_Confirm ("Move Event to "+String(SX\_StartTm;5)  
  +"-"+String(SX\_EndTm;5)+"?")

**Else**

```
$bo_AllowAction:=Action_Confirm ("Move "+t_oldTag+" to "+$t_newTag  
+" at "+String(SX_StartTm;5)+"-"+String(SX_EndTm;5)+"?")
```

```
End if
```

```
End if
```

```
End if `bPrompt=1
```

```
: (SX_Action=SX_kResizeBefore) `Save the Start and End Times to be used in the "Resize After" action.
```

```
c_oldStartTime:=SX_StartTm
```

```
c_oldEndTime:=SX_EndTm
```

```
If (i_CurrentView=1) `Dates View
```

```
    If (SX_PKey<0)
```

```
        $bo_AllowAction:=False `should be Lunch Event; don't allow moving
```

```
    End if
```

```
End if
```

```
: (SX_Action=SX_kResizeAfter)
```

```
    If (bPrompt=1)
```

```
        $bo_AllowAction:=Action_Confirm ("Change from "+String(c_oldStartTime;5)  
        +"-"+String(c_oldEndTime;5)+" to "+String(SX_StartTm;5)+"-"  
        +String(SX_EndTm;5)+"?")
```

```
    End if
```

```
: (SX_Action=SX_kEditBefore)
```

```
: (SX_Action=SX_kEditAfter)
```

```
: (SX_Action=SX_kColorBefore)
```

```
: (SX_Action=SX_kColorAfter)
```

```
: (SX_Action=SX_kDelete)
```

```
    $bo_AllowAction:=Action_Confirm ("Are you sure you want to do this?")
```

**: (SX\_Action=SX\_kArraysBefore)**

```

If (i_CurrentView=1) `Dates View
    $bo_AllowAction:=DoArraysBeforeAction
End if

```

**: (SX\_Action=SX\_kArraysAfter)**

```

If (i_CurrentView=1) `Dates View
    $bo_AllowAction:=DoArraysAfterAction
End if

```

**: (SX\_Action=SX\_kContextualBefore) `New in v4**

```

ARRAY TEXT(SX_APopupMenuItems;0;0)

```

```

If (SX_PKey#0)
    QUERY([Activity];[Activity]ID=SX_PKey)
    If (Records in selection([Activity])=1)
        Build_ContextualMenu
        UNLOAD RECORD([Activity])
    End if
Else
    Build_ContextualMenu2
End if

```

```

` if the User chooses a developer menu item these variables will be set to the row &
` column of the 'SX_APopupMenuItems' array on a 'SX_kContextualAfter' action
SX_PopupDevRow:=0
SX_PopupDevCol:=0

```

**: (SX\_Action=SX\_kContextualAfter) `New in v4**

```

If (SX_PKey#0)

```

```

` if the User chooses a developer menu item the 'SX_PopupDevRow' & 'SX_PopupDevCol'
` variables will be set to the corresponding element of the 'SX_APopupMenuItems' array

```

```

If (SX_PopupDevRow>0)

QUERY([Activity];[Activity]ID=SX_PKey)
If (Records in selection([Activity])=1)

Case of

```

```
      : (SX_PopupDevRow=row_Color_i)
        C_INTEGER($ok)
        C_LONGINT($chosenColor)
        PLATFORM PROPERTIES($platform)

        If ($platform=Windows )
            $chosenColor:=SXIndexToColor (SX_PopupDevCol-1)
            $ok:=1
        Else
            $ok:=SXColorPicker ([Activity]BG_Color;$chosenColor)
        End if

        If ($ok=1)
            [Activity]BG_Color:=$chosenColor
        End if

      : (SX_PopupDevRow=row_Therapists_i)
        [Activity]ID_Therapist_FK:=al_Thrpst_PKey{SX_PopupDevCol}

      : (SX_PopupDevRow=row_Rooms_i)
        [Activity]ID_Room_FK:=al_Room_PKey{SX_PopupDevCol}

      : (SX_PopupDevRow=row_Courses_i)
        [Activity]ID_Course_FK:=al_Class_PKey{SX_PopupDevCol}

      : (SX_PopupDevRow=row_Equip_i)
        [Activity]ID_Equipment_FK:=al_Equip_PKey{SX_PopupDevCol}

    End case

    SAVE RECORD([Activity])
    UNLOAD RECORD([Activity])

    SX_Action:=SX_kDoReload

  End if

End if

Else `This leg handles the end-user clicking on a one month calendar date open area

  If (SX_PopupDevRow=row_Color_i)
    C_INTEGER($ok)
    C_LONGINT($chosenColor)

    PLATFORM PROPERTIES($platform)

    If ($platform=Windows )
```

```

        $chosenColor:=SXIndexToColor (SX_PopupDevCol-1)
        $ok:=1
    Else
        $ok:=SXColorPicker (16777212;$chosenColor)
        `Establish neutral color in center of wheel as starting point
    End if

    If ($ok=1)

        $result:=SXGetView (E_SCHED;$ViewKind)
        If ($ViewKind=2) `The Month View
            $DayNumber:=Day of(SX_ClickDate)
            $errResult:=SXMonthSetDayColor (E_SCHED;
                $DayNumber;$chosenColor)
            ` set chosen day color from color wheel
        End if
    End if
End if

End if

: (SX_Action=SX_kOnDrop) `New in v4

If (i_CurrentView=1) ` Dates or Month View

$err:=SXGetDroppedObject (E_SCHED;$source;$arrayrow;$processnum;$destDate;
    $destResourceID;$destStartTime;$destEndTime)
    ` get information on the source & destination of the drop
If ($err=0)

    If ($source=Get pointer("textVar1")) ` source was from the 'textVar1' variable on the form

        If (SX_PKey#0)
            QUERY([Activity];[Activity]ID=SX_PKey)
            If (Records in selection([Activity])=1)
                [Activity]Summary:=textVar1
                SAVE RECORD([Activity])
                UNLOAD RECORD([Activity])
                SX_Action:=SX_kDoReload
            End if
        Else
            End if
    End if

```

**End if**  
**End if**  
**End if**

: (SX\_Action=SX\_kOnTip) `New in v4

SX\_TipText:=String(SX\_StartTm;HH MM AM PM )+"-"+String(SX\_EndTm;HH MM AM PM )+"  
"+SX\_Summary

**End case**

\$0:=\$bo\_AllowAction

## The Chart Commands

**'SetupChart(ChartArea)**

**C\_LONGINT(\$1;\$ChartArea;\$l\_error;\$c\_StartTime;\$c\_EndTime)**

**C\_INTEGER(\$Number\_Therapists)**

\$ChartArea:=\$1

`Setup column header options

`SXChartColHdrOpt(L;I;I;S;I;I;I)

`L -> Area

`I -> Column Width (-1 = no change, 0 = auto, 1 = scale to fit, >5 = fixed)

`I -> Header height (-1 = no change, 0 = auto, >5 = fixed points)

`S -> Font name ("" = no change)

`I -> Font size (-1 = no change)

`I -> Font style (-1 = no change)

`I -> Font color (-1 = no change)

`I -> Background color (-1 = no change)

\$l\_error:=SXChartColHdrOpt (\$ChartArea;100;0;"Palatino";12;1;1;5)

`Setup resource label options

`SXChartResourcesOpt(L;I;I;S;I;I;I)

`L -> Area

`I -> Header width (-1 = no change, 0 = auto, >5 = fixed points)

`I -> Row height (-1 = no change, 0 = auto, 1 = scale to fit, >5 = fixed)

`S -> Font name ("" = no change)

`I -> Font size (-1 = no change)

`I -> Font style (-1 = no change)

`I -> Font color (-1 = no change)  
 `I -> Background color (-1 = no change)

\$!\_error:=SXChartResourcesOpt (\$ChartArea;0;60;"Helvetica";18;0;1;5) `Purple;200)

`Setup grid options  
 `SXChartGridOpt(L;I;I;I;I;I;I)  
 `L -> Area  
 `I -> Horizontal lines visible (-1 = no change, 0 = no, 1 = yes)  
 `I -> Horizontal lines size (-1 = no change)  
 `I -> Horizontal lines color (-1 = no change)  
 `I -> Vertical lines visible (-1 = no change, 0 = no, 1 = yes)  
 `I -> Vertical lines size (-1 = no change)  
 `I -> Vertical lines color (-1 = no change)  
 `I -> Background color (-1 = no change)

\$!\_error:=SXChartGridOpt (\$ChartArea;1;1;Light Grey;1;1;Dark Grey;Light Blue)

`Setup column headers  
 ARRAY TEXT(at\_ColumnHeaders;11)  
 \$c\_StartTime:=†08:00:00†-0 ` -0 so 4D respects this variable as a long integer  
 \$c\_EndTime:=†18:00:00†-0  
 at\_ColumnHeaders{1}:= "8AM"  
 at\_ColumnHeaders{2}:= "9AM"  
 at\_ColumnHeaders{3}:= "10AM"  
 at\_ColumnHeaders{4}:= "11AM"  
 at\_ColumnHeaders{5}:= "12PM"  
 at\_ColumnHeaders{6}:= "1PM"  
 at\_ColumnHeaders{7}:= "2PM"  
 at\_ColumnHeaders{8}:= "3PM"  
 at\_ColumnHeaders{9}:= "4PM"  
 at\_ColumnHeaders{10}:= "5PM"  
 at\_ColumnHeaders{11}:= "6PM"

`SXChartColumns(&L;&I;&X;&L;&L)  
 `L -> Area  
 `I -> Alignment ( 0 = center over divider, 1 = center in column )  
 `X -> Headers (array of text)  
 `L -> Start value  
 `L -> End value

\$!\_error:=SXChartColumns (\$ChartArea;0;at\_ColumnHeaders;\$c\_StartTime;\$c\_EndTime)

## Chart\_Resource\_Setup

```
`Chart_Resource_Setup
```

```
ARRAY TEXT($Resources;0)
```

```
C_LONGINT($l_error)
```

```
ARRAY LONGINT(al_Key_ID;0)
```

```
C_INTEGER($i_Number_Records)
```

### Case of

```
: (as_ViewChoice=1)
```

```
  ALL RECORDS([Therapist])
```

```
  ORDER BY([Therapist];[Therapist]Therapist_Name;>)
```

```
  SELECTION TO ARRAY([Therapist]Therapist_Name;$Resources;[Therapist]Therapist_PKey;al_Key_ID)
```

```
: (as_ViewChoice=2)
```

```
  ALL RECORDS([Class])
```

```
  ORDER BY([Class];[Class]Course_Name;>)
```

```
  SELECTION TO ARRAY([Class]Course_Name;$Resources;[Class]Course_PKey;al_Key_ID)
```

```
: (as_ViewChoice=3)
```

```
  ALL RECORDS([Room])
```

```
  ORDER BY([Room];[Room]Room_Name;>)
```

```
  SELECTION TO ARRAY([Room]Room_Name;$Resources;[Room]Room_PKey;al_Key_ID)
```

```
: (as_ViewChoice=4)
```

```
  ALL RECORDS([Equipment])
```

```
  ORDER BY([Equipment];[Equipment]Equip_Type;>)
```

```
  SELECTION TO ARRAY([Equipment]Equip_Type;$Resources;[Equipment]Equip_PKey;al_Key_ID)
```

```
: (as_ViewChoice=5)
```

```
  ALL RECORDS([Client])
```

```
  ORDER BY([Client];[Client]Client_LName;>)
```

```
  SELECTION TO ARRAY([Client]Client_Name;$Resources;[Client]Client_PKey;al_Key_ID)
```

### End case

```
$i_Number_Records:=Size of array($Resources)
```

```
` $l_error:=SXChartResources(&L;&T;&X)
```

```
  `L -> Area
```

```
  `T -> Header
```

```
  `X -> Resources (array of text)
```

\$l\_error:=SXChartResources (\$ChartArea;"People";\$Resources)

**ARRAY LONGINT**(\$al\_Starts;0)

**ARRAY LONGINT**(\$al\_Ends;0)

**ARRAY INTEGER**(\$ai\_Colors;0)

**For** (\$i;1;\$i\_Number\_Records)

QUERY([Activity];[Activity]Act\_SDate=d\_beginDate;\*) `Replace with Calendar date picked

**Case of**

: (as\_ViewChoice=1)

QUERY([Activity]; & ;[Activity]Therapist\_FKey=al\_Key\_ID{\$i})

: (as\_ViewChoice=2)

QUERY([Activity]; & ;[Activity]Course\_FKey=al\_Key\_ID{\$i})

: (as\_ViewChoice=3)

QUERY([Activity]; & ;[Activity]Room\_FKey=al\_Key\_ID{\$i})

: (as\_ViewChoice=4)

QUERY([Activity]; & ;[Activity]Equipment\_FKey=al\_Key\_ID{\$i})

: (as\_ViewChoice=5)

QUERY([Activity]; & ;[Activity]Client\_FKey=al\_Key\_ID{\$i})

**End case**

SELECTION TO ARRAY([Activity]Act\_STime;\$al\_Starts;[Activity]Act\_ETime;\$al\_Ends;[Activity]Act\_Color\_Index;\$ai\_Colors)

` \$l\_error:=SXChartSetResourceValues(&L;&L;&X;&X;&X)

` L -> Area

` L -> Resource index

` X -> Start values (array of longint)

` X -> End values (array of longint)

` X -> Colors (array of integer)

**For** (\$j;1;Records in selection([Activity]))

\$l\_error:=SXChartSetResourceValues (\$ChartArea;\$i;\$al\_Starts;\$al\_Ends;\$ai\_Colors)

**End for**

**End for**

## Color Assignment Options

Using the Control key, click on the object whose color you wish to reassign. The Macintosh OS X color palette and the 4D color palette options are available on the Macintosh. Developers can specify which color palette they want to display using the appropriate commands. will initially display the color currently selected. By moving the mouse pointer to the desired color and releasing the mouse, the object/event is automatically reassigned the selected color. The Mac OS X color wheel stores its numeric value in a long integer and the 4D color palette stores its color in an integer field.

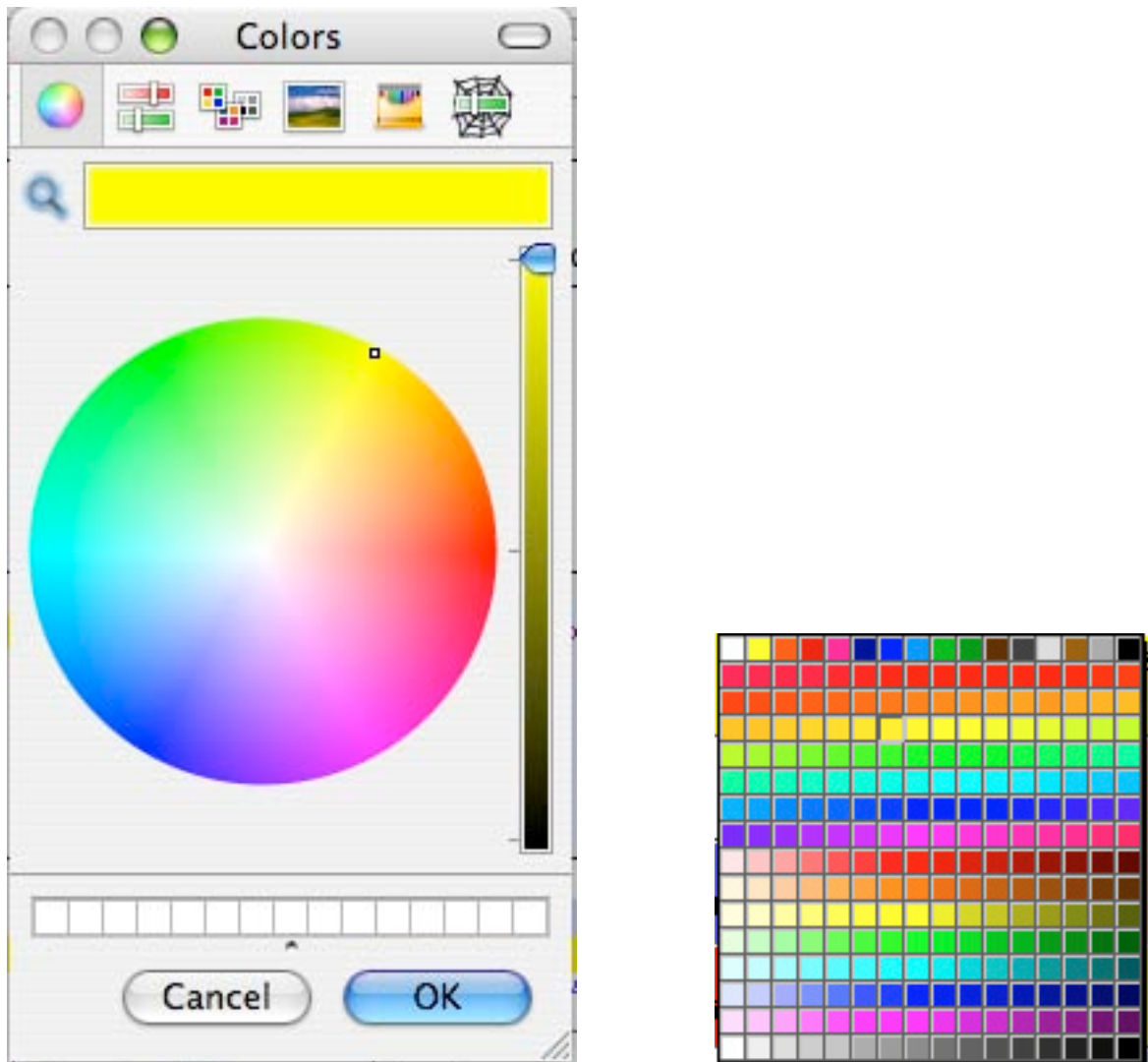


Figure 10. The OS X Color Wheel and 4D Color Grid

## The Calendar plug-in Area

March 1999						
Sun	Mon	Tue	Wed	Thu	Fri	Sat
	1	2	3	4	5	6
7	8	9	10	11	12	13
14	15	16	17	18	19	20
21	22	23	24	25	26	27
28	29	30	31			

Figure 11. The Calendar plug-in Area

When the calendar area is clicked on, a script calls the **ChangeStartDate** method as follows:

```
\ChangeStartDate(ScheduleArea) (abbreviated)
```

```
\The Start Date has changed.
```

```
\Show all Events with the new date.
```

```
C_LONGINT ($1;$e_Sched;$l_error)
```

```
C_STRING (80;$s_WindowTitle)
```

```
$e_Sched:=$1
```

```
d_endDate:=d_beginDate+RulerIntervals
```

### Case of

```
: (i_CurrentView=1) \Dates View
```

```
  $l_error:=SXShowDates ($e_Sched;d_beginDate;d_endDate;al_Thrpst_PKey)
```

```
  $s_WindowTitle:=String(d_beginDate;3)+" - "+String(d_endDate;3)
```

```
: (i_CurrentView=2) \Therapists View
```

```
  $l_error:=SXShowResources ($e_Sched;d_beginDate;al_Thrpst_PKey)
```

```
  $s_WindowTitle:="Therapists "+String(d_beginDate;3)
```

### Else

```
  $l_error:=0
```

```
  $s_WindowTitle:=""
```

### End case

```
If ($l_error=0)
```

```
  SET WINDOW TITLE ($s_WindowTitle)
```

```
Else
```

```
  Error_Alert ($l_error)
```

```
End if
```

## The Calendar Area Script

`[Table1]SX.E\_CAL

**C\_LONGINT** (\$l\_error)

**C\_DATE** (\$d\_beginDate)

### Case of

: (**Form event**=On Plug in Area)

\$l\_error:=*SXCalGetDate* (E\_CAL;\$d\_beginDate)

**If** (\$l\_error=0)

d\_beginDate:=\$d\_beginDate

*ChangeStartDate* (E\_SCHED)

**Else**

*Error\_Alert* (\$l\_error)

**End if**

*Set\_Cal\_Day\_Color*

**End case**

**SchedulePack Release History:**

<b>4.0</b>	<ul style="list-style-type: none"> <li>- Add 4D v11 compatibility,</li> <li>- Introduce the one-month view and the time-line view.</li> <li>- Integrate all features and capabilities of the one-month, the time-line and the Resource/Date views within a single SchedulePack external area.</li> <li>- Three new activity fields added for tighter iCal integration and compatibility.</li> <li>- Added an enhanced time-line, supporting multiple contiguous days, developer definable hash-marks, end-user interaction via double click, contextual menus, gradient colors, including all other features available to other areas.</li> <li>- Built-in contextual menus with program callback access.</li> <li>- Creation of a single area to support all six calendar areas; Resource, Date, Timeline, Monthly, Weekly &amp; Daily views.</li> <li>- 5 Additional callback neumonics added to support new feature set.</li> <li>- Added picture support directly within events and banners.</li> <li>- Quartz and gradient color support (Macintosh).</li> <li>- Full Mac OS X color picker support.</li> <li>- iCal export and import capabilities.</li> <li>- Drag and drop support for any 4D-based objects, including 3<sup>rd</sup> part plug-in objects.</li> <li>- Added 4D Advanced Properties dialog support.</li> <li>- Optimized internal search algorithm for event queries.</li> </ul>
<b>3.0.8</b>	<ul style="list-style-type: none"> <li>- SXEventOpt; Added parameter supporting round activity rectangles.</li> <li>- SXRowHdrFont; Added support for a user-definable row height.</li> <li>- SXLabelsTable; Added support for a new row header color option; based on related resource values.</li> </ul>
<b>3.0.7</b>	<ul style="list-style-type: none"> <li>- the nagging alert for an expired deployment no longer displays with compiled DB.</li> </ul>
<b>3.0.6-Mac</b>	<ul style="list-style-type: none"> <li>- Mac Only - fixed occasional problem where scrollbars would not draw correctly.</li> </ul>
<b>3.0.5</b>	<p>New License code checking logic:</p> <ul style="list-style-type: none"> <li>- database running as Client/Engine (compiled) - check deployment code only</li> <li>- database running as Client/Engine (uncompiled) - check deployment code; if invalid then check developer code</li> <li>- database running in any other 4D environment - check developer code only</li> </ul>
<b>3.0.4</b>	<p>Fixed double-click problem that occurred on MacOS X. The callback was getting a "move" action instead of a "double-click" action.</p>
<b>3.0.3</b>	<p>Fixed double-click problem on overlapping events. After double-clicking, the same summary text would sometimes display in both events.</p>
<b>3.0.2</b>	<p>New deployment scheme for the following cases:</p> <ul style="list-style-type: none"> <li>- Compiled Database, 4D Client - check deployment code only</li> <li>- Compiled Database, 4D Engine - check deployment code only</li> <li>- Compiled Database, all other 4D - check developer code only</li> <li>- Uncompiled Database, 4D Client - check developer &amp; deployment codes</li> <li>- Uncompiled Database, 4D Engine - check developer &amp; deployment codes</li> <li>- Uncompiled Database, all other 4D - check developer code only</li> </ul>
<b>3.0.1</b>	<p>Will now check the deployment code for a 4D Engine license when running in that environment.</p>

## Index

- 4D
  - Developer Methods, 5
- Action Codes, 65
  - SXkArraysAfter, 81
  - SXkColorAfter, 81
  - SXkColorBefore, 81
  - SXkContextualAfter, 81
  - SXkContextualBefore, 81
  - SXkDelete, 81
  - SXkDoRefresh, 81
  - SXkDoReload, 81
  - SXkDoubleClick, 81
  - SXkEditAfter, 81
  - SXkEditBefore, 81
  - SXkMoveAfter, 80
  - SXkMoveBefore, 80
  - SXkNewAfter, 80
  - SXkNewArea, 80
  - SXkNewBefore, 80
  - SXkNewEdit, 80
  - SXkOnDrop, 81
  - SXkQueryBefore, 81
  - SXkResizeAfter, 80
  - SXkResizeBefore, 80
- Action Codes Reference, 80
- Activation
  - SXRegister, 76
  - SXShowDates, 77
  - SXShowResources, 77
- Activation Reference, 76
- Banner Events, 20
- Banners Reference, 77
- Calendar Area, 41, 108
- Calendar Control
  - SXCalClear, 78
  - SXCalDateColor, 78
  - SXCalDateIcon, 78
  - SXCalFonts, 78
  - SXCalFormat, 78
  - SXCalGetDate, 79
  - SXCalGetMonth, 78
  - SXCalGetYear, 78
  - SXCalSetDate, 79
  - SXCalSetMonthYr, 79
- Calendar Control Reference, 78, 79
- Calendar Navigation, 76
- Callback Support Reference, 80
- Callback Variables, 62
  - SX\_Action, 63, 80
  - SX\_ClickDt, 65
  - SX\_ClickEndTime, 65
  - SX\_ClickStartTime, 65
  - SX\_ColorIdx, 63, 64, 80
  - SX\_CTagFKey, 63, 80
  - SX\_EndDt, 62, 80
  - SX\_EndTm, 63, 80
  - SX\_ETagFKey, 63, 80
  - SX\_EvtType, 64, 80
  - SX\_Param1, 64, 80
  - SX\_PKey, 62, 80
  - SX\_StartDt, 62, 80
  - SX\_StartTm, 62, 80
  - SX\_Summary, 63, 80
- Chart Control Reference, 79
- Coloring Events, 76
- Configuration
  - SXCallbackProc, 77
  - SXColumnsTable, 77
  - SXEventsTable, 77
  - SXGetScroll, 77
  - SXLabelsTable, 77
  - SXRefresh, 78
  - SXSetScroll, 78
  - SXTimeIntervals, 77
- Configuration Reference, 77
- Creating Events, 75
- Deleting Events, 76
- Developer Methods, 5
- Equivalent Integer Value, 80
- Error Codes, 6, 7, 28, 30, 31, 35, 36, 37, 38, 39, 40, 44, 45, 46, 47, 81
- Events
  - Coloring, 76
  - Creating, 75
  - Deleting, 76
  - Moving, 76
  - Resizing, 76
- Incoming Action Codes, 66
  - SXkArraysAfter, 73
  - SXkArraysBefore, 71
  - SXkColorAfter, 70
  - SXkColorBefore, 69
  - SXkDelete, 70
  - SXkDoubleClick, 70
  - SXkEditAfter, 69
  - SXkEditBefore, 69
  - SXkMoveAfter, 68
  - SXkMoveBefore, 67
  - SXkNewAfter, 66
  - SXkNewArea, 66

- SXkNewBefore, 66
- SXkNewEdit, 67
- SXkResizeAfter, 68
- SXkResizeBefore, 68
- Mnemonic, 80
- Moving Events, 76
- Outgoing Action Codes, 75**
  - SXkContextualAfter, 64, 73**
  - SXkContextualBefore, 73
  - SXkDoRefresh, 75
  - SXkDoReload, 75
  - SXkOnDrop, 74
- Preferences
  - SXAreaOpt, 77
  - SXBannerRow, 77
  - SXColHdrFont, 77
  - SXDateFormat, 77
  - SXDragCreateOpt, 77
  - SXDragMoveOpt, 77
  - SXDragResizeOpt, 77
  - SXEventFont, 77
  - SXEventOpt, 77
  - SXEvtLabelFont, 77
  - SXMinColWidth, 77
  - SXRowHdrFont, 77
  - SXSetColor, 77
  - SXTimeFormat, 77
- Preferences Reference, 77
- Resizing Events, 76
- SX\_Action, 63, 66, 67, 68, 69, 70, 71, 72, 73, 74, 75, 80
- SX\_ClickDt, 65
- SX\_ClickEndTime, 65
- SX\_ClickStartTime, 65
- SX\_ColorIdx, 63, 64, 80
- SX\_CTagFKey, 63, 80
- SX\_EndDt, 62, 80
- SX\_EndTm, 63, 67, 80
- SX\_ETagFKey, 63, 67, 80
- SX\_EvtType, 64, 80
- SX\_Param1, 64, 80
- SX\_PKey, 31, 62, 67, 70, 80
- SX\_RecordNo, 67
- SX\_StartDt, 62, 80
- SX\_StartTm, 62, 67, 80
- SX\_Summary, 63, 80
- SXAreaOpt, 13, 14, 77
  - Description, 13
  - Example, 14
  - Parameters, 13
  - Syntax, 13
- SXBannerRow, 20, 77
  - Description, 20
  - Example, 20
  - Parameters, 20
  - Syntax, 20
- SXBanners, 4**
- SXCalClear, 45, 78
  - Description, 45
  - Error Codes, 45
  - Example, 45
  - Parameters, 45
  - Syntax, 45
- SXCalDateColor, 43, 44, 45, 78
  - Description, 43
  - Error Codes, 44
  - Example, 44
  - Parameters, 44
  - Syntax, 44
- SXCalDateIcon, 44, 45, 78
  - Description, 44
  - Error Codes, 45
  - Example, 45
  - Parameters, 44
  - Syntax, 44
- SXCalendar, 41
- SXCalFonts, 43, 78
  - Description, 43
  - Example, 43
  - Parameters, 43
  - Syntax, 43
- SXCalFormat, 42, 43, 78
  - Description, 42
  - Example, 43
  - Parameters, 42
  - Syntax, 42
- SXCalGetDate, 47, 79
  - Description, 47
  - Error Codes, 47
  - Example, 47
  - Parameters, 47
  - Syntax, 47
- SXCalGetMonth, 45, 46, 78
  - Description, 45
  - Error Codes, 46
  - Example, 46
  - Parameters, 46
  - Syntax, 45
- SXCalGetYear, 46, 78
  - Description, 46
  - Error Codes, 46
  - Example, 46
  - Parameters, 46
  - Syntax, 46
- SXCallbackProc, 4, 23, 77**
  - Description, 23
  - Example, 23
  - Parameters, 23
  - Syntax, 23
- SXCalSetDate, 48, 79
  - Description, 48
  - Example, 48
  - Parameters, 48

- Syntax, 48
- SXCalSetMonthYr, 46, 47, 48, 79
  - Description, 46
  - Example, 47
  - Parameters, 47
  - Syntax, 47
- SXColHdrFont, 8, 9, 77
  - Description, 8
  - Example, 8, 9
  - Parameters, 8
  - Syntax, 8
- SXColorPicker, 32
  - Description, 32
  - Example, 32
  - Parameters, 32
  - Return Value, 32
  - Syntax, 32
- SXColorPopup, 31, 78
  - Description, 31
  - Example, 31
  - Parameters, 31
  - Return Value, 31
  - Syntax, 31
- SXColorToIndex, 33
  - Description, 33
  - Example, 33
  - Parameters, 33
  - Return Value, 33
  - Syntax, 33
- SXColorToRGB, 33, 34
  - Description, 33
  - Example, 34
  - Parameters, 34
  - Return Value, 34
  - Syntax, 34
- SXColumnsTable**, 4, 24, 27, 28, 77
  - Description, 27
  - Error Codes, 28
  - Example, 28
  - Parameters, 28
  - Syntax, 28
- SXDateFormat, 11, 12, 77
  - Description, 11
  - Example, 12
  - Parameters, 11
  - Syntax, 11
- SXDeleteEvent, 30, 31, 78
  - Description, 30
  - Error Codes, 31
  - Example, 31
  - Parameters, 30
  - Syntax, 30
- SXDragCreateOpt, 15, 16, 77
  - Description, 15
  - Example, 16
  - Parameters, 15
  - Syntax, 15
- SXDragMoveOpt, 16, 17, 77
  - Description, 16
  - Example, 17
  - Parameters, 16
  - Syntax, 16
- SXDragResizeOpt, 17, 77
  - Description, 17
  - Example, 17
  - Parameters, 17
  - Syntax, 17
- SXEventFont, 9, 77
  - Description, 9
  - Example, 9
  - Parameters, 9
  - Syntax, 9
- SXEventOpt, 14, 15, 77
  - Description, 14
  - Example, 15
  - Parameters, 14
  - Syntax, 14
- SXEventsTable**, 4, 24, 26, 28, 77, 86
  - Description, 24
  - Error Codes, 26
  - Example, 26
  - Parameters, 24
  - Syntax, 24
- SXEvtLabelFont, 10, 18, 77
  - Description, 10, 18
  - Example, 10, 18
  - Parameters, 10, 18
  - Syntax, 10, 18
- SXExport, 39, 40
  - Description, 39
  - Error Codes, 40
  - Example, 40
  - Parameters, 40
  - Syntax, 40
- SXGetDraggedEvent, 38, 40
  - Description, 38, 40
  - Error Codes, 38, 40
  - Example, 38, 40
  - Parameters, 38, 40
  - Syntax, 38, 40
- SXGetDroppedObject, 38, 39
  - Description, 38
  - Error Codes, 39
  - Example, 39
  - Parameters, 38
  - Syntax, 38
- SXGetScroll, 36, 77
  - Description, 36
  - Example, 36
  - Parameters, 36
  - Syntax, 36
- SXGetSelected, 30, 78

- Description, 30
- Error Codes, 30
- Example, 30
- Parameters, 30
- Syntax, 30
- SXGetView**, 29
- SXIndexToColor**, 33
  - Description, 33
  - Example, 33
  - Parameters, 33
  - Return Value, 33
  - Syntax, 33
- SXkArraysAfter**, 73
  - Description, 73
  - Result, 73
  - Syntax, 73
- SXkArraysBefore**, 71, 72
  - Description, 71
  - Result, 72
  - Syntax, 72
- SXkColorAfter**, 70
  - Description, 70
  - Result, 70
  - Syntax, 70
- SXkColorBefore**, 69, 70
  - Description, 69
  - Result, 70
  - Syntax, 70
- SXkContextualAfter**, 64, 73
  - Description, 73
  - Result, 73
  - Syntax, 73
- SXkContextualBefore**, 73
  - Description, 73
  - Result, 73
  - Syntax, 73
- SXkDelete**, 70
  - Description, 70
  - Result, 70
  - Syntax, 70
- SXkDoRefresh**, 75
  - Description, 75
  - Result, 75
  - Syntax, 75
- SXkDoReload**, 75
  - Description, 75
  - Result, 75
  - Syntax, 75
- SXkDoubleClick**, 70, 71
  - Description, 70
  - Result, 71
  - Syntax, 71
- SXkEditAfter**, 69
  - Description, 69
  - Result, 69
  - Syntax, 69
- SXkEditBefore**, 69
  - Description, 69
  - Result, 69
  - Syntax, 69
- SXkMoveAfter**, 68
  - Description, 68
  - Result, 68
  - Syntax, 68
- SXkMoveBefore**, 67, 68
  - Description, 67
  - Result, 68
  - Syntax, 68
- SXkNewAfter**, 66, 67
  - Description, 66
  - Result, 67
  - Syntax, 67
- SXkNewArea**, 66
  - Description, 66
  - Result, 66
  - Syntax, 66
- SXkNewBefore**, 66
  - Description, 66
  - Result, 66
  - Syntax, 66
- SXkNewEdit**, 67
  - Description, 67
  - Result, 67
  - Syntax, 67
- SXkOnDrop**, 74, 75
  - Description, 74
  - Result, 74, 75
  - Syntax, 74, 75
- SXkQueryBefore**, 71
- SXkResizeAfter**, 68, 69
  - Description, 68
  - Result, 69
  - Syntax, 69
- SXkResizeBefore**, 68
  - Description, 68
  - Result, 68
  - Syntax, 68
- SXLabelsTable**, 4, 24, 26, 27, 77
  - Description, 26
  - Error Codes, 27
  - Example, 27
  - Parameters, 26
  - Syntax, 26
- SXMinColWidth**, 10, 11, 77
  - Description, 10
  - Example, 11
  - Parameters, 11
  - Syntax, 11
- SXMonthClearPictures**, 50
- SXMonthFont**, 18, 19, 49, 50
- SXMonthGetMonthYear**, 51
- SXMonthGetMonthYear**, 51

- SXMonthOpt, 49
- SXMonthOpt, 49
- SXMonthSetColor, 51
- SXMonthSetColor, 51
- SXMonthSetDatePicture, 52
- SXMonthSetDatePicture, 52
- SXMonthSetDayColor, 52
- SXMonthSetDayColor, 52
- SXMonthSetDayColor, 54
- SXMonthSetMonthYear, 53
- SXMonthSetMonthYear, 53
- SXRefresh, 37, 78
  - Description, 37
  - Error Codes, 37
  - Example, 37
  - Parameters, 37
  - Syntax, 37
- SXRegister, 5, 76
  - Description, 5
  - Example, 5
  - Parameters, 5
  - Syntax, 5
- SXRGBToColor, 34
  - Description, 34
  - Example, 34
  - Parameters, 34
  - Return Value, 34
  - Syntax, 34
- SXRowHdrFont, 7, 8, 77
  - Description, 7
  - Example, 8
  - Parameters, 7
  - Syntax, 7
- SXRowToTime, 35, 78
  - Description, 35
  - Error Codes, 35
  - Example, 35
  - Parameters, 35
  - Syntax, 35
- SXSetColor, 12, 13, 77
  - Description, 12
  - Example, 13
  - Syntax, 12
- SXSetScroll, 36, 37, 78
  - Description, 36
  - Error Codes, 37
  - Example, 37
  - Parameters, 36
  - Syntax, 36
- SXSetView, 29
- SXShowDates**, 4, 5, 6, 28, 77
  - Description, 5
  - Error Codes, 6
  - Example, 6
  - Parameters, 6
  - Syntax, 6
- SXShowResources**, 4, 6, 7, 27, 77
  - Description, 6
  - Error Codes, 7
  - Example, 7
  - Parameters, 7
  - Syntax, 6
- SXTimeFormat, 12, 77
  - Description, 12
  - Example, 12
  - Parameters, 12
  - Syntax, 12
- SXTimeIntervals, 22, 23, 77
  - Description, 22
  - Error Codes, 18, 19, 23
  - Example, 23
  - Parameters, 22
  - Syntax, 22
- SXTimeLineFont**, 4, 54, 55
- SXTimeLineFont, 55
- SXTimeLineOpt**, 4, 54
- SXTimeLineSetColor**, 4, 55
- SXTimeLineSetColor, 55
- SXTimeToRow, 35, 36, 78
  - Description, 35
  - Error Codes, 36
  - Example, 36
  - Parameters, 35
  - Syntax, 35
- Tabbing, 76
- Timeline Commands, 53
- Timeline Control Reference, 79
- Tips, 75
- Utilities
  - SXColorPopup, 78
  - SXDeleteEvent, 78
  - SXGetSelected, 78
  - SXRowToTime, 78
  - SXTimeToRow, 78
- Utilities Reference, 78