

SchedulePack™

Version 3.04



by Soft Solutions

© 1996-2003 *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



Results Through Focused Technology...

IMPORTANT—READ CAREFULLY BEFORE OPENING. By installing this software program, 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, non-transferable 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 an unlimited number of custom or 4D-compiled non-commercial database applications created by the original licensee. No additional product license or royalty is required for non-commercial database applications.

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 which 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, decimate, 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 Compiler, 4D External Mover and 4D are registered trademarks of 4D, Inc.

Table of Contents

Introduction	1
Simplifying the Complex	1
Solving Common Business Problems	1
Introducing the Time-Line	1
Other Supported Features	1
Compatibility	2
Overview	3
Command Usage and Typical Order:	3
Which Plug-in Do I Use?	4
What's New in v3?	5
The Time-Line and Charting Tool	5
Custom Searching and Display	5
Complete Overlap Control	6
Breaking the Midnight Barrier	6
4D Developer API routines	7
Preferences	7
SXRegister	7
SXRowHdrFont	8
SXColHdrFont	9
SXEventFont	10
SXEvtLabelFont	11
SXMinColWidth	12
SXTimeFormat	12
SXDateFormat	13
SXAreaOpt	14
SXEventOpt	15
SXDragCreateOpt	16
SXDragResizeOpt	16
SXDragMoveOpt	17
Banner Events	18
SXBannerRow	19

Configuration	19
SXTimeIntervals	19
SXCallbackProc	21
SXEventsTable	21
SXLabelsTable	23
SXColumnsTable	25
Activation	26
SXShowDates	26
SXShowResources	27
SXGetScroll	27
SXSetScroll	28
SXRefresh	28
Utilities	29
SXGetSelected	29
SXDeleteEvent	30
SXColorPopup	30
SXRowToTime	31
SXTimeToRow	32
Calendar Control Routines	33
The SchedulePack Calendar Area	33
SXCalFormat	33
SXCalFonts	34
SXCalDateColor	35
SXCalDateIcon	35
SXCalClear	36
SXCalGetMonth	37
SXCalGetYear	37
SXCalSetMonthYr	38
SXCalGetDate	38
SXCalSetDate	39
Chart Commands	40
The SchedulePack Chart Commands	40
SXChartColumns	40
SXChartResources	41
SXChartSetResourceValues	41
SXChartColHdrOpt	42
SXChartResourcesOpt	43
SXChartGridOpt	43
SXChartAreaOpt	44
Callback Support	45
Overview	45

Callback Variables	45
SX_PKey	45
SX_StartDt	46
SX_EndDt	46
SX_StartTm	46
SX_EndTm	46
SX_Summary	46
SX_ColorIdx	46
SX_CTagFKey	47
SX_ETagFKey	47
SX_Action	47
SX_EvtType	47
SX_Param1	47
Action Codes	48
Incoming Action Codes	48
SXkNewArea	48
SXkNewBefore	48
SXkNewAfter	49
SXkNewEdit	50
SXkMoveBefore	50
SXkMoveAfter	51
SXkResizeBefore	51
SXkResizeAfter	51
SXkEditBefore	52
SXkEditAfter	52
SXkColorBefore	52
SXkColorAfter	53
SXkDelete	53
SXkDoubleClick	53
SXkQueryBefore	54
SXkArraysBefore	54
SXkArraysAfter	56
Outgoing Action Codes	56
SXkDoReload	56
SXkDoRefresh	56
User Tips	58
Creating Events/Objects	58
Deleting Events/Objects	58
Moving Events/Objects	58
Resizing Events/Objects	58
Coloring Events/Objects	58

Tabbing	58
Calendar Navigation	58
Command Summary:	59
Preferences	59
Configuration	59
Utilities	60
Calendar Control	60
Chart Control	60
CallBack Support Variables	61
Action Codes	62
Error Codes	63
Sample Application	67
SchedulePack Definition and Setup	67
Menu System for Demo Example	68
Typical Command Sequence	68
Application Definition and Setup	69
Declaration of Arrays for Related Resource Tables	70
Callback Variable Declaration	71
Action Variable Declaration	72
SchedulePack Area Form Method	73
Callback Examples	77
The Chart Commands	82
‘SetupChart(ChartArea)	82
Chart_Resource_Setup	83
Color Assignment	85
The Calendar plug-in Area	86
The Calendar Area Script	88

Introduction

SchedulePack is a 4th Dimension plug-in providing compelling graphical display of date and time events along with their associated resources. SchedulePack provides compatibility for 4D 6.8 and 4D 2003, running under Macintosh OS 8, 9 and X and Windows.

Simplifying the Complex

SchedulePack provides an attractive drag and drop interface and a friendly user environment that adds significant informational value to any 4D application. Program simplification of common and complex scheduling tasks makes SchedulePack an ideal centerpiece for scheduling applications.

With SchedulePack, the myriad of complex resource possibilities for any time period can be fully managed with minimal 4D programming. Historically, programming only a portion of these combinations was tedious, labor intensive and difficult to maintain. SchedulePack provides a new level of sophistication without the associated programming headaches. Developers will appreciate how tough scheduling assignments are now made easy with SchedulePack.

Companies choosing to program a software scheduling solution with pure 4D code are faced with an underlying complexity and quickly realize how hard it is to simplify complex tasks. Consequently, many automated 4D scheduling solutions either end up as incomplete, inadequate, difficult to use and difficult to maintain. SchedulePack eliminates this dilemma.

Solving Common Business Problems

Scheduling solutions are needed in schools, medical offices, businesses, and countless organizations who require resources, such as people, rooms, etc. to be reserved for certain time windows on a certain day. Common to educational institutions is a class whose resources may include professors, students, equipment, rooms, etc. In a medical office or hospital setting, doctors, patients, rooms and assistants are all resources which must be continuously monitored and managed to ensure the business is effectively operated. At a sports complex, playing fields, officials, scorekeepers and participants may all have to be carefully managed to ensure resources are properly and uniquely allocated at any one time. Scheduled events may have mutually exclusive resources, whose proper allocation is essential to the successful scheduling. 4D Developers can add appropriate rules and criteria to filter out unavailable resources. This simple and intuitive resource allocation increases SchedulePack's value, especially as scheduling problems grow more complex.

Introducing the Time-Line

The v3 charting tool is a separate plug-in area and an excellent tool for creating time-line solutions. 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 all employee work hours along with any scheduling conflicts in color. In addition, the time-line area doubles as a viable charting tool.

Other Supported Features

The Events or Activities tables represent the primary currency of the SchedulePack plug-in. SchedulePack can use 4D records and/or array-based solutions. This avoids extra Developer steps and provides for quick implementation of robust scheduling features and applications. From its 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 all be handled directly by SchedulePack with-

out any programming. Alternatively, callbacks can be used in conjunction with a complete set of Action Codes or mnemonics created from User-initiated actions. Through callbacks, SchedulePack provides the capability for complete customizing.

SchedulePack was designed to work with an “Events” table, where the specific “default” table name is defined internally. For this reason, developers can name this table and its 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 which they intend to display.

SchedulePack's appearance and configuration are fully customizable under version 3. 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 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 Calendar plug-in is also provided 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.

Multi-day banner events are supported. Through this capability, End Users can visually manage their time and ensure Event conflicts do not exist. Full callback support also allows the Developer to customize the application specific to the scheduling requirements.

SchedulePack can be displayed as an independent, resizable external window and will take full advantage of larger monitors.

Compatibility

SchedulePack 3.0 requires 4th Dimension 6.0 or higher and includes full compatibility with 4th Dimension v6.8 and 4D 2003. SchedulePack is also compatible with all corresponding version levels of 4D Server. SchedulePack requires the Macintosh Operating System 8.0 or above, Windows 95 or greater and Windows NT version 3.5 or greater.

Overview

The SchedulePack version 3 plug-in provide 3 separate plug-in areas. A calendar area, for simple date selection and display, a Scheduling area, rich with callback capabilities providing in-depth developer control and a new charting area, providing customizable display and print capabilities. The charting tool, introduced in version 3, is ideal for providing end-users with a valuable time-line 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 variables which are 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.

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 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).

Which Plug-in Do I Use?

Table 1:

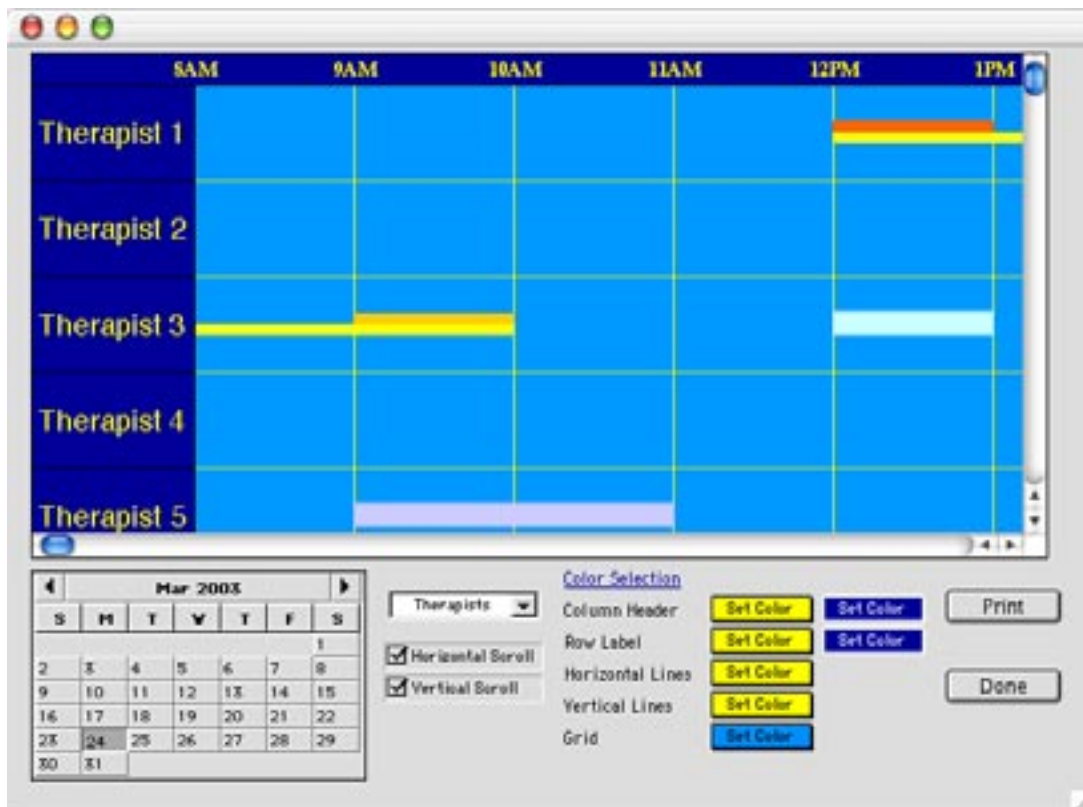
	Mac OS 9.2 or earlier	Mac OS X or later	Windows - All
4D 6.8 or later	Mac4DX or Mac4CX	Mac4CX	Win4DX
4D 6.7 or earlier	Mac4DX	N/A	Win4DX

What's New in v3?

SchedulePack version 3 provides Macintosh OS X and 4th Dimension 2003 compatibility.

The Time-Line and Charting Tool

The SchedulePack version 3 plug-in adds a new charting tool, providing 4D developers customizable display and print capabilities ideal for providing end-users with a valuable time-line display. There are seven related charting commands, providing a detailed level of programmer control over custom charts and time-lines.



Custom Searching and Display

A new callback action is available in version 3 called the SXkQueryBefore action. This special callback capability was introduced to allow 4D developers full control over the SchedulePack display area contents by directly populating the SchedulePack arrays according to their applications 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-times 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.

Complete Overlap Control

Version 3 handles the display and accessibility of multiple event overlaps more equitably. Prior to version 3, busy time slots became easily overcrowded with SchedulePack event objects. Under version 3, 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.

Breaking the Midnight Barrier

In version 3, the SXTimeIntervals command now accepts both time and integer values, allowing the SchedulePack 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 who need to manage scheduling from one day over into the next.

4D Developer API routines

Preferences

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.

The "INVALID DEPLOYMENT" message appears along with the serial number of the 4D Server where the current 4D application is running. This 4D Server serial number is required in order to obtain an individual 4D server deployment number from Soft Solutions. SchedulePack will automatically present the current 4D Server serial number when a 4D User attempts to run on 4D Client/Server without a valid deployment license number. Alternatively, the 4D Developer can obtain this serial number directly by issuing the 4D command "Get Serial Information".

An OEM deployment license is based on the Licensee name and does not require an individual 4D Server serial number. 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. 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 SSL.

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))

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

Example:

SXRowHdrFont(xSch;"Geneva";14;1) '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

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))
```

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
```

Example:

```
SXEventFont(xSch;"Monaco";12;0) `12 point Monaco Plain
```

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

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

Example:

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

SXEvtLabelFont

Description:

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

Syntax:

```
SXEvtLabelFont(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

Example:

```
SXEvtLabelFont(xSch;"Monaco";12;2)
```

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

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

Example:

```
SXEvtLabelFont(xArea;"";12;4)`don't change Font,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
```

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.

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 column-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 Form Method”](#) on page 73.

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))
```

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.

Example:

```
SXEventOpt(xSch;1;1;1;1;12)
```

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)
```

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)

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.

CanDragBannerAn 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.

CanDragVertEvtAn 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)

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 Form Method”](#) on page 73.

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:=SXTimeIntervals(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 69.

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 - Integer
- 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 data-types 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 23](#) and [SXCColumnsTable on page 25](#) 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))->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 for each event.

The color index is a number between 0 and 255 which identifies the color from the application's color palette which is used to fill in the event box when it is drawn on the grid. The field's type must be **Integer**.

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 an 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**.

Error Codes:

- 0 - Successful call made
- 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 Index Field Type
- 15009 - Invalid Event Type Field Type

Example:

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

To see an example, please refer to the [“Application Definition and Setup”](#) on page 69.

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);EventLblIDFld(I);LblTableNo(I);LblTableIDFld(I);LblTableStrFld(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.

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.

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

Example:

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

To see an example, please refer to the [“Application Definition and Setup”](#) on page 69.

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.

ColTableNoAn 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 *LblTableNo*) 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 69.

Activation

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)

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)

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)

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)
```

Calendar Control Routines

The SchedulePack Calendar Area

SchedulePack 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 API routines” on page 7 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 Form Method”](#) on page 73.

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 Form Method”](#) on page 73.

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 Form Method”](#) on page 73.

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 Form Method”](#) on page 73.

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 Form Method”](#) on page 73.

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 **SXCalSetDate** 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;1998) 'Display the first month of 1998
```

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 Form Method”](#) on page 73.

Chart Commands

The SchedulePack Chart Commands

In version 3, SchedulePack contains a new, additional 4D plug-in area providing charting and time-line capabilities. This plug-in area can be used to provide users with a convenient way to view and print the daily time allocations for any number of resources (Users, rooms, equipment, classes, etc.) .

This chart area is ideal for timelines and 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.

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";\$Resources)

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 TimesAn 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;al_StartTime;al_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;I;S;I;I;I;I)

SXChartColHdrOpt(xArea(L);ColumnWidth(I);HeaderHeight(I);FontName(S);FontSize(I);FontStyle(I);FontColor(I);Background Color(I))

Parameters:

xAreaPass 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).

FontNameAn string expression specifying the font name (a null value for no change).

FontSize.....An integer expression specifying the font size (-1 = no change).

FontStyleAn integer expression specifying the font style (-1 = no change).

FontColorAn 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);FontStyle(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 BarAn integer expression specifying to show or hide the vertical scroll bars (-1 = no change, 0=hide, 1 =show).

Example:

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

Callback Support

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 77](#).

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 77](#).

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 77](#).

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 77](#).

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 77](#).

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 77](#).

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.

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_ACIdx" - 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" and "SX_AEvtText" 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

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 imple-

mented 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:

Preferences

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

SXRowHdrFont(xArea(L);FontName(S);FontSize(I);FontStyle(I))

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

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

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

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

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

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

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

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

SXEventOpt(xArea(L);CanMakeBnr(I);CanColorBnr(I);CanMakeEvt(I);CanColorEvt(I);OverlapIndent(I))

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

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

SXDragMoveOpt(xArea(L);CanDragBnr(I);CanDragEvtHoriz(I);CanDragEvtVert(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))->L

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

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

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

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

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

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

SXRefresh(xArea(L))->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

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))

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

Action Codes

Action Mnemonic	Equivalent Integer Value
SXkNewArea	-1
SXkNewBefore	1
SXkNewAfter	2
SXkNewEdit	3
SXkMoveBefore	10
SXkMoveAfter	11
SXkResizeBefore	20
SXkResizeAfter	21
SXkEditBefore	30
SXkEditAfter	31
SXkColorBefore	40
SXkColorAfter	41
SXkDelete	51
SXkDoubleClick	52
SXkQueryBefore	120
SXkArraysAfter	121
SXkDoReload	101
SXkDoRefresh	102

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 Index Field Type

Solution: Change the Color Index field to field type Integer.

-15009 - Invalid Event Type Field Type

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

-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.

-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.

-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.

-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 SXTimeInterval command times the number of intervals per hour. A number larger than this would generate the -15601 error code.

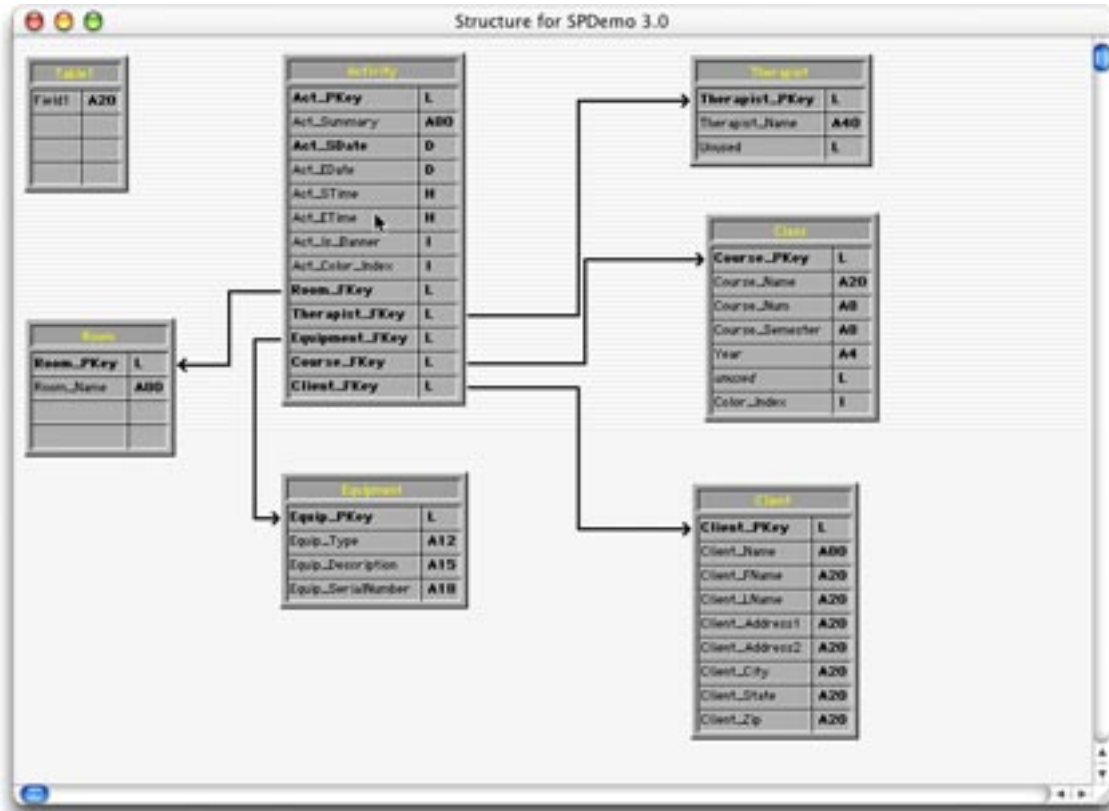
-15602 - Attempt to Delete unknown record

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

Sample Application

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.



SchedulePack Definition and Setup

Within your 4D application, you’ll be required to define all forms or dialogs which include the SchedulePack area. After drawing the SchedulePack area on the form, you can double-click on the area and define the SchedulePack area as follows:



It is up to the Developer to provide a name to the SchedulePack area. In the above example, the area has been defined as “E_Sched”.

Within the demo applications menu system, each menu item invokes a method called *DO_DemoX*, where x contains the relative value of the menuitem number.

Menu System for Demo Example



For each of the first 6 menu items is a 4D method (or procedure) named “Do_DemoX” where “x” corresponds to the menu item number shown above (see keyboard equivalent). Each “Do_DemoX” method calls the Test Method passing it an integer with the value corresponding to “x”. Such as Do_Demo1 would contain the following line of code: *Do_Demo(1)*. The *Do_Demo* method initializes arrays, sets the default Columns and Labels Tables, specifies the SchedulePack callback procedure (only for menu item 5) and displays the window.

Typical Command Sequence

Although there are approximately two dozen schedule related commands which provide for 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, you must first declare (as in 4D compiler declarations) the SchedulePack variables.** These variables are found in the “Callback Variables” on page 45 of this manual. Failure to do so may yield unpredictable values and possibly crash the machine, especially when working with source code. The essential SchedulePack commands and flow is 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 specify whether or not a banner row is to be shown. 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 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 "Command Reference:" on page 51.

Application Definition and Setup

```
`Do_Demo(DemoMode)

C_LONGINT ($1;l_demoMode)
C_INTEGER (i_CurrentView) `Current View - Dates, Therapists, Classes, Rooms, Equipment,
or Clients
C_INTEGER (i_DefaultColor) `Default color for New Events
C_BOOLEAN (bo_HasCallBack) `True if the Schedule Area has a callback method
C_BOOLEAN (bo_LunchEvents) `True if Lunch Events are to be created in Dates View
C_DATE (d_beginDate) `Start Date in Dates View; one specific Date for all other Views
C_DATE (d_endDate) `End Date in Dates View
C_LONGINT (l_RoomFKKey) `ID of Room Record to show in Dates View
C_LONGINT ($l_error)

l_demoMode:=$1

bo_HasCallBack:=(l_demoMode=5)

Declare_Vars

i_DefaultColor:=Yellow

If (l_demoMode=5)
  i_CurrentView:=1 `Dates View
Else
  i_CurrentView:=l_demoMode
End if

l_RoomFKKey:=-1 `default to show all Rooms
bo_LunchEvents:=False `don't create Lunch Events

d_beginDate:=Current date
d_endDate:=d_beginDate

`Build the Resource Arrays
Therapist_To_Array
Class_To_Array
Room_To_Array
Equipment_To_Array
Client_To_Array
```

```
If (bo_HasCallBack)
    `install a default callback method; New Schedule Areas will use this method
    SXCallbackProc (0;"SPack_Callback")
Else
    SXCallbackProc (0;"")
End if
```

```
`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_Color_Index);Field(->[Activity]Act_Is_Banner))
```

```
If ($!_error # 0)
    Error_Alert ($!_error)
```

```
End if
```

```
New_SPACK_Window `Display Window with Calendar & Schedule
```

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_PKey;[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)

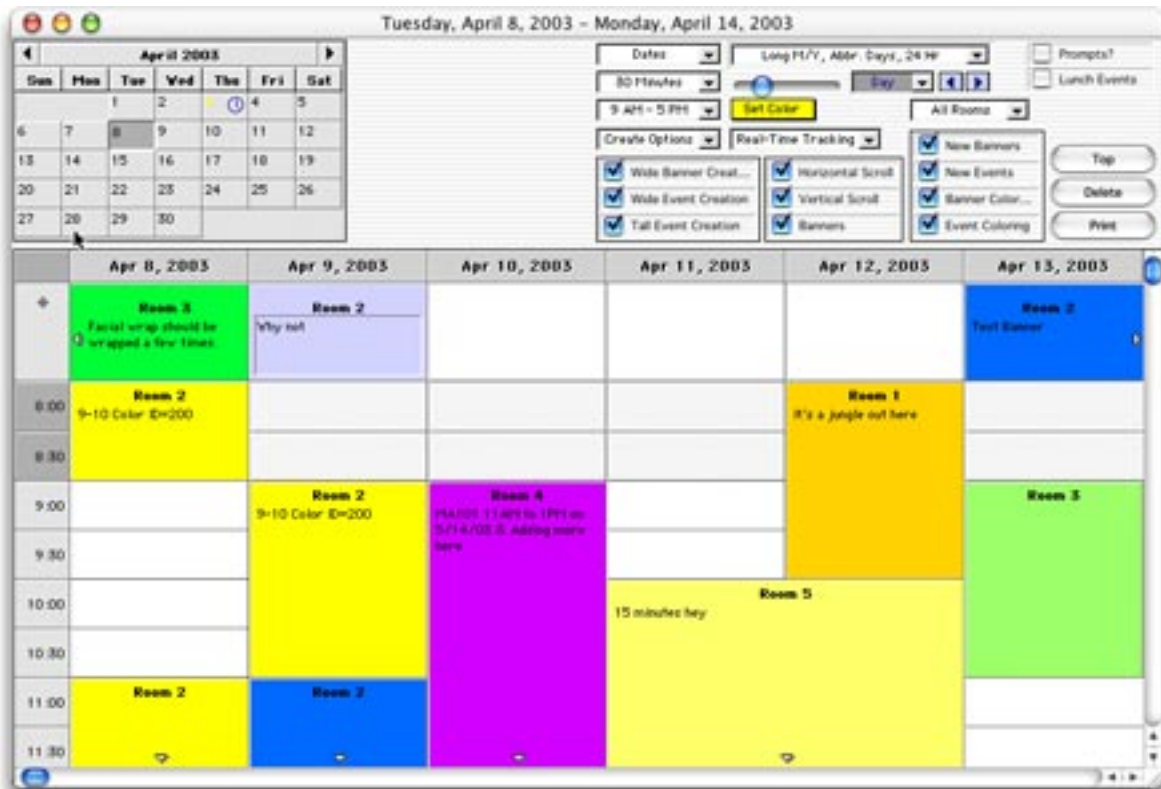
Action Variable Declaration

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 Form Method



`[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

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") `Initialize the 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}:="6 AM - 12 PM"
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)

as_Pop_Color{1}:="Set Color"

as_Pop_Color:=1

Date/Time Format Choices

ARRAY STRING (30;as_dateTimeFmt;3)

as_dateTimeFmt{1}:="Long M/Y, Abbr. Days, 24 Hr"

as_dateTimeFmt{2}:="Short M/Y, Short Day, 12 Hr"

as_dateTimeFmt{3}:="Short M/Y, Long Day, v1.0 Time"

as_dateTimeFmt:=1 `Default to "Long M/Y, Abbr. Days, 24 Hr" choice

Event Tracking Choices

ARRAY STRING (20;as_Tracking;2)

as_Tracking{1}:="Real-Time Tracking"

as_Tracking{2}:="Outline Tracking"

as_Tracking:=1

If (bo_HasCallBack) `has a callback method

l_RoomFKey:=-1 `flag to show all Rooms

COPY ARRAY (as_RoomTag;as_Rooms)

INSERT ELEMENT (as_Rooms;1)

as_Rooms{1}:="All Rooms"

as_Rooms:=1

Set Lunch Events Checkbox

If (bo_LunchEvents)

b_LunchEvents:=1

Else

b_LunchEvents:=0

End if

End if `bo_HasCallBack

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 Horizontal Scrollbar, Show Vertical Scroll Bar,
Real_Time Tracking

SXEventOpt (E_SCHED;1;1;1;1) `Create New Banners, Color Banners, Create New Events,
Color Events

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_kNewBefore)

: (SX_Action=SX_kDoubleClick) `Double-Clicked on an Event

 ` "SX_PKey" contains the Record ID of the Event

 Edit_Event (SX_PKey)

 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 the Record ID of the Event
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 `should be Lunch Event; don't allow moving
  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
  $l_elemNum:=Find in array(al_Room_PKey;SX_CTagFKKey)
  If ($l_elemNum>0)
    t_oldTag:=as_RoomTag{$l_elemNum}
  End if

```

```

: (i_CurrentView=5) `Equipment View
  $l_elemNum:=Find in array(al_Equip_PKey;SX_CTagFKKey)
  If ($l_elemNum>0)
    t_oldTag:=as_EquipTag{$l_elemNum}
  End if

```

```

: (i_CurrentView=6) `Clients View
  $l_elemNum:=Find in array(al_Client_PKey;SX_CTagFKKey)
  If ($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
  $l_elemNum:=Find in array(al_Class_PKey;SX_CTagFKey)
  If ($l_elemNum>0)
    $t_newTag:=as_ClassNum{$l_elemNum}
  End if
```

```
: (i_CurrentView=4) `Rooms View
  $l_elemNum:=Find in array(al_Room_PKey;SX_CTagFKey)
  If ($l_elemNum>0)
    $t_newTag:=as_RoomTag{$l_elemNum}
  End if
```

```
: (i_CurrentView=5) `Equipment View
  $l_elemNum:=Find in array(al_Equip_PKey;SX_CTagFKey)
  If ($l_elemNum>0)
    $t_newTag:=as_EquipTag{$l_elemNum}
  End if
```

```
: (i_CurrentView=6) `Clients View
  $l_elemNum:=Find in array(al_Client_PKey;SX_CTagFKey)
  If ($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

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)

\$l_error:=SXChartResourcesOpt (\$ChartArea;0;60;"Helvetica";18;0;1;5) `Purple;200)

`Setup grid options

`SXChartGridOpt(L;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)

```
$l_error:=SXChartGridOpt ($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
```

```
$l_error:=SXChartColumns ($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;[Equip-
ment]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;$al_Starts;[Activ-
ity]Act_ETime;$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

Using the Control key, click on the object whose color you wish to reassign. The color palette 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 Calendar plug-in Area



When the calendar area is clicked on, a script calls the **ChangeStartDate** method as follows:

```
\ChangeStartDate(ScheduleArea)
```

```
\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)

: (i_CurrentView=3) `Classes View
  $l_error:=SXShowResources ($e_Sched;d_beginDate;al_Class_PKey)
  $s_WindowTitle:="Classes "+String(d_beginDate;3)

: (i_CurrentView=4) `Rooms View
  $l_error:=SXShowResources ($e_Sched;d_beginDate;al_Room_PKey)
  $s_WindowTitle:="Rooms "+String(d_beginDate;3)

: (i_CurrentView=5) `Equipment View
  $l_error:=SXShowResources ($e_Sched;d_beginDate;al_Equip_PKey)
  $s_WindowTitle:="Equipment "+String(d_beginDate;3)

: (i_CurrentView=6) `Clients View
  $l_error:=SXShowResources ($e_Sched;d_beginDate;al_Client_PKey)
  $s_WindowTitle:="Clients "+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

Numerics

4D

API routines 7

A

Action Codes 48

- SXkArraysAfter 62
- SXkArraysBefore 62
- SXkColorAfter 62
- SXkColorBefore 62
- SXkDelete 62
- SXkDoRefresh 62
- SXkDoReload 62
- SXkDoubleClick 62
- SXkEditAfter 62
- SXkEditBefore 62
- SXkMoveBefore 62
- SXkNewAfter 62
- SXkNewArea 62
- SXkNewBefore 62
- SXkNewEdit 62
- SXkResizeAfter 62
- SXkResizeBefore 62

Action Codes Reference 62

API routines 7

B

Banner Events 18

C

Calendar Area 33, 40

Calendar Control

- SXCalClear 60
- SXCalDateColor 60
- SXCalDateIcon 60
- SXCalFonts 60
- SXCalGetDate 60
- SXCalGetMonth 60
- SXCalGetYear 60
- SXCalSetDate 60
- SXCalSetMonthYear 60

Calendar Control Reference 60

Calendar Navigation 58

Callback Support 45

Callback Support Reference 61

Callback Variables 45

- SX_Action 47, 61
- SX_ColorIdx 46, 61
- SX_CTagFKey 47, 61
- SX_EndDt 46, 61
- SX_EndTm 46, 61
- SX_ETagFKey 47, 61
- SX_EvtType 47
- SX_Param1 47
- SX_PKey 45, 61
- SX_StartDt 46, 61
- SX_StartTm 46, 61
- SX_Summary 46, 61

Coloring Events 58

Configuration

- SXCallbackProc 59
- SXColumnsTable 59
- SXEventsTable 59
- SXGetScroll 59
- SXLabelsTable 59
- SXRefresh 59
- SXSetScroll 59
- SXShowDates 59
- SXShowResources 59
- SXTimeIntervals 59

Configuration Reference 59

Creating Events 58

D

Deleting Events 58

E

Equivalent Integer Value 62

Error Codes

24, 25, 26, 27, 28, 29, 30, 31,
32, 35, 36, 37, 38

Events

- Coloring 58
- Creating 58
- Deleting 58
- Moving 58
- Resizing 58

I

Incoming Action Codes 48

- SXkArraysAfter 56
- SXkArraysBefore 54
- SXkColorAfter 53
- SXkColorBefore 52
- SXkDelete 53
- SXkDoubleClick 53
- SXkEditAfter 52
- SXkEditBefore 52
- SXkMoveAfter 51
- SXkMoveBefore 50
- SXkNewAfter 49
- SXkNewArea 48
- SXkNewBefore 48
- SXkNewEdit 50
- SXkResizeAfter 51
- SXkResizeBefore 51

M

Mnemonic 62

Moving Events 58

O

Outgoing Action Codes 56

- SXkDoRefresh 56
- SXkDoReload 56

P

Preferences

- SXBannerRow 59
- SXColHdrFont 59
- SXDateFormat 59
- SXEvtLabelFont 59
- SXEvtTextFont 59
- SXMinColWidth 59
- SXRowHdrFont 59

Preferences Reference 59

R

Resizing Events 58

S

SX_Action 47, 48, 49, 51, 52, 53, 54, 56, 61

SX_ColorIdx 46, 61

SX_CTagFKey 47, 61

SX_EndDt 46, 61

SX_EndTm 46, 49, 61

SX_ETagFKey 47, 49, 61

SX_EvtType 47

SX_Param1 47

SX_PKey 29, 45, 49, 53, 61

SX_RecordNo 50

SX_StartDt 46, 61

SX_StartTm 46, 49, 61

SX_Summary 46, 61

SXAreaOpt 14, 59

Description 14

Example 14

Syntax 14

SXAreaOpts

Parameters 14

SXBannerRow 19, 59

Description 19

Example 19

Parameters 19

Syntax 19

SXCalClear 36, 60

Description 36

Error Codes 36

Example 36

Parameters 36

Syntax 36

SXCalDateColor 35, 36, 60

Description 35

Error Codes 35

Example 35

Parameters 35

Syntax 35

SXCalDateIcon 35, 36, 60

Description 35

Error Codes 36

Example 36

Parameters 35

Syntax 35

SXCalendar 33

SXCalFonts 34, 42, 43, 60

Description 34, 40, 41, 42, 43, 44

Example 34, 40, 41, 42, 43,

44, 45

Parameters 34, 40, 41, 42, 43, 44, 45

Syntax 33, 34, 40, 41, 42, 43, 44

SXCalFormat 33, 40, 41, 44, 60

Description 33

Example 34

Parameters 33

SXCalGetDate 38, 60

Description 38

Error Codes 38

Example 38

Parameters 38

Syntax 38

SXCalGetMonth 37, 60

Description 37

Error Codes 37

Example 37

Parameters 37

Syntax 37

SXCalGetYear 37, 60

Description 37

Error Codes 37

Example 37

Parameters 37

Syntax 37

SXCallbackProc 21, 59

Description 21

Example 21

Parameters 21

Syntax 21

SXCalSetDate 38, 39, 60

Description 39

Example 39

Parameters 39

Syntax 39

SXCalSetMonthYr 38, 39, 60

Description 38

Example 38

Parameters 38

Syntax 38

SXColHdrFont 9, 59

Description 9

Example 9

Parameters 9

Syntax 9

SXColorPopup 30, 60

Description 30

Example 31

Parameters 31

Return Value 31

Syntax 31

SXColumnsTable 22, 25, 27, 59

Description 25

Error Codes 25

Example 26

Parameters 25

Syntax 25

SXDateFormat 13, 59

Description 13

Example 13

Parameters 13

Syntax 13

SXDeleteEvent 30, 60

Description 30

Error Codes 30

Example 30

Parameters 30

Syntax 30

SXDragCreateOpt 16, 59

Description 16

Example 16

Parameters 16

Syntax 16

SXDragMoveOpt 17, 59

Description 17

Example 18

Parameters 17

Syntax 17

SXDragResizeOpt 16, 59

Description 16

Example 17

Parameters 17

Syntax 16

SXEventFont 10

Description 10

Example 10

Parameters 10

Syntax 10

SXEventOpt 15, 59

Description 15

Example 15

Parameters 15

Syntax 15

SXEventsTable 21, 24, 25, 59

Description 21

Error Codes 23

Example 23

Parameters 22

Syntax 22

SXEvtLabelFont 11, 59

Description 11

Example 11

Parameters 11

Syntax 11

- SXEvtTextFont 59
 - SXGetScroll 27, 59
 - Description 27
 - Example 28
 - Parameters 27
 - Syntax 27
 - SXGetSelected 29, 60
 - Description 29
 - Error Codes 29
 - Example 30
 - Parameters 29
 - Syntax 29
 - SXkArraysAfter 56, 62
 - SXkArraysBefore 54, 62
 - SXkColorAfter 53, 62
 - Description 53
 - Result 53
 - Syntax 53
 - SXkColorBefore 52, 62
 - Description 52
 - Result 52
 - Syntax 52
 - SXkDelete 53, 62
 - Description 53
 - Result 53
 - Syntax 53
 - SXkDoRefresh 56, 62
 - Description 56
 - Result 57
 - Syntax 57
 - SXkDoReload 56, 62
 - Description 56
 - Result 56
 - Syntax 56
 - SXkDoubleClick 53, 62
 - Description 53
 - Result 54
 - Syntax 54
 - SXkEditAfter 52, 62
 - Description 52
 - Result 52
 - Syntax 52
 - SXkEditBefore 52, 62
 - Description 52
 - Result 52
 - Syntax 52
 - SXkMoveAfter 51
 - Description 51
 - Result 51
 - Syntax 51
 - SXkMoveBefore 50, 62
 - Description 50
 - Result 50
 - Syntax 50
 - SXkNewAfter 49, 62
 - Description 49
 - Result 49
 - Syntax 49
 - SXkNewArea 48, 62
 - Description 48
 - Result 48, 49
 - Syntax 48, 49
 - SXkNewBefore 48, 62
 - SXkNewEdit 49, 50, 62
 - Description 50
 - Result 50
 - Syntax 50
 - SXkResizeAfter 51, 62
 - Description 51
 - Result 51
 - Syntax 51
 - SXkResizeBefore 51, 62
 - Description 51
 - Result 51
 - Syntax 51
 - SXLabelsTable 22, 23, 59
 - Description 23
 - Error Codes 24, 25, 26, 27, 28, 29, 30, 31, 32, 35, 36, 37, 38
 - Example 24, 25, 26, 27, 28, 29, 30, 31, 32, 35, 36, 37, 38
 - Parameters 24
 - Syntax 24
 - SXMinColWidth 12, 59
 - Description 12
 - Example 12
 - Parameters 12
 - Syntax 12
 - SXRefresh 28, 59
 - Description 28
 - Error Codes 29
 - Example 29
 - Parameters 29
 - Syntax 29
 - SXRegister 7, 59
 - Description 7
 - Example 8
 - Parameters 7
 - Syntax 7
 - SXRowHdrFont 8, 59
 - Description 8
 - Example 9
 - Parameters 8
 - Syntax 8
 - SXRowToTime 31, 60
 - Description 31
 - Error Codes 31
 - Example 31
 - Parameters 31
 - Syntax 31
 - SXSetScroll 27, 28, 59
 - Description 28
 - Error Codes 28
 - Example 28
 - Parameters 28
 - Syntax 28
 - SXShowDates 24, 25, 26, 59
 - Description 26
 - Error Codes 26
 - Example 26
 - Parameters 26
 - Syntax 26
 - SXShowResources 24, 25, 27, 59
 - Description 27
 - Error Codes 27
 - Parameters 27
 - Syntax 27
 - SXTimeFormat 12, 59
 - Description 12
 - Example 13
 - Parameters 12
 - Syntax 12
 - SXTimeIntervals 19, 59
 - Description 19
 - Error Codes 20
 - Example 20
 - Parameters 20
 - Syntax 20
 - SXTimeToRow 32, 60
 - Description 32
 - Error Codes 32
 - Example 32
 - Parameters 32
 - Syntax 32
- T**
- Tabbing 58
 - Tips 58
- U**
- Utilities
 - SXColorPopup 60
 - SXDeleteEvent 60
 - SXGetSelected 60
 - SXRowToTime 60
 - SXTimeToRow 60
 - Utilities Reference 60