

Using SureTrak Scripts

In this chapter:

Using Basic Scripts
SureTrak Objects,
Functions, and
Properties

Scripts are programs that tell an application what to do. Use them to automate tasks that you do repetitively or regularly. Scripts can be simple or complex, depending on the complexity of the tasks you want to automate.

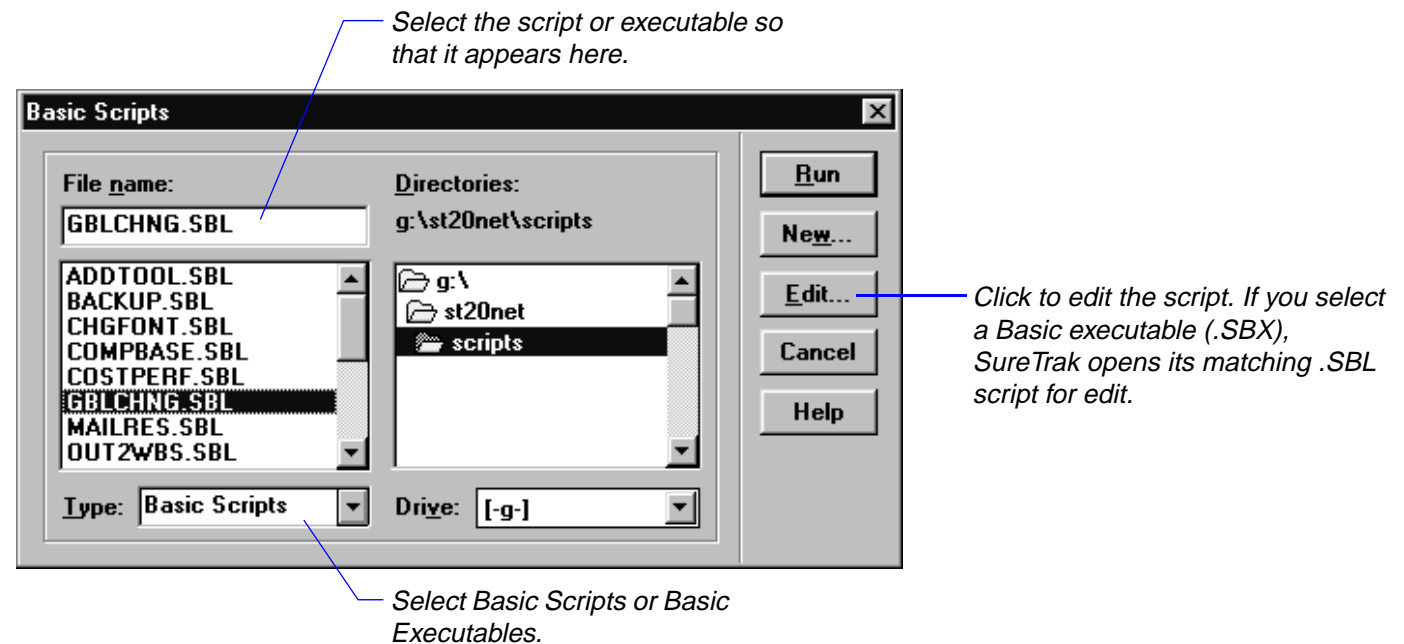
SureTrak includes several sample SBL scripts that you can use to automate daily tasks in SureTrak.

Using Basic Scripts

SureTrak includes two kinds of Basic programs: scripts, which have an .SBL extension; and executables, which have an .SBX extension. Nearly all .SBX files have matching .SBL scripts. You can run or edit the .SBL scripts included with SureTrak, or write your own scripts. You can run the .SBX programs, but you cannot edit them directly.



To use a Basic script from SureTrak, choose Tools, Basic Scripts (or click the Basic Scripts icon), specify whether you want to use a script or an executable, then click Run.



You can add a Basic executable to the SureTrak Custom Tools menu for easy access.

To add a Basic executable to the SureTrak Custom Tools menu

- 1 Open the STWIN.INI file with Notepad.
- 2 Add a section called [TOOLMENU], followed by a line with the menu option and program filename you want to run. For example,

```
[TOOLMENU]
&Global Change=c:\stwin\scripts\gblchnng.sbx
```

- 3 To run the Basic executable, choose Custom Tools, then the program name.

You can also use the Basic program called ADDTOOL to add scripts to the Custom Tools menu automatically.

SureTrak's Basic Scripts SureTrak includes 17 Basic scripts described in the following table:

Script	Function
ADDTOOL	Updates SureTrak's Custom Tools menu, adding .SBX scripts or .EXE programs that you specify.
BACKUP	Triggers an ON_PROJECT_CLOSE event, creates a BACKUP directory below the directory containing the project, and then backs up the project being closed. If the project files were not saved before the ON_PROJECT_CLOSE event occurred, the project is saved before it is backed up.
CHGFONT	Changes the font used in the current layout.
COMPBASE	Exports or imports a comma-separated (.CSV) file containing activity ID, budgeted cost (optional), and start and finish dates (specify target, early, late, or leveled dates).
COSTPERF	Calculates resource To Complete costs based on CPI index (earned value/actual cost) and saves the values to project files.
GBLCHNG	Performs one of four functions, either globally or based on a filter: updates durations or percent complete; deletes activities; modifies resource assignments; modifies resource and cost assignments.

Script	Function
MAILRES	Creates an INI setting for resource E-mail addresses.
OUT2WBS	Copies the outline codes to the WBS code field.
PREDSUCC	Creates a predecessor/successor report.
PRNCODES	Enables you to print your Resource, Activity Code, and WBS Code Dictionaries.
REINDEX	Deletes all index files for a SureTrak-type project.
RESASGN	Creates a Resource Assignment report.
RESCOST	Creates a Resource Cost report.
SPELLCHK	Copies information to Excel and performs spell checking using OLE automation.
XCONST	Writes external constraints to activity logs.
XLRESCST	Creates a Resource Cost Report in Excel using OLE automation.
ZER2MIL	Converts zero-duration activities to start milestones.

You can edit any scripts that are delivered with SureTrak to customize them for your own use.

To edit an existing Basic script



- 1 Click the Basic Scripts icon, or choose Tools, Basic Scripts to open the Basic Scripts dialog box.
- 2 Select Basic Scripts or Basic Executables in the Type field.
- 3 Select the script you want to edit.
- 4 Click Edit.
- 5 Change the Basic script as appropriate.

The editor functions like the standard Windows editor: you can type text directly using the keyboard or copy it from the Clipboard.

- 6 Choose File, Save or File, Save As to save the changed script.

To write a new Basic Script



- 1 Click the Basic Scripts icon, or choose Tools, Basic Scripts to open the Basic Scripts dialog box.

- 2 Click New.

If most of your scripts begin with the same definitions and inclusions, you can specify a default script as a template. In the [Features] section of the STWIN.INI file, specify Blank Script = <script name>.

- 3 Write the new Basic script in the editor.

- 4 Save the new script.

The Editing and Debugging Environment When you click New or Edit in the Basic Scripts dialog box, SureTrak displays the BASIC Language dialog box. Use the editing and debugging controls on the toolbar to modify the script.

```

*****
* Copyright 1997 Primavera Systems, Inc.
*
*
* GBLCHNG.SBL - Globally or based on a filter does one of the
* following based on user selection
* a.) Updates durations or percent complete
* b.) Deletes activities
* c.) Modifies resource assignments
* d.) Modifies resource and cost assignments
*
*****\






//***** SureTrak Header Files *****/
$include "prmconst.sbh"
$include "prmcmd.sbh"
$include "gblchng.sbh"
$include "wincmd.sbh"
//*****













Option Explicit                                '// Catches undefined variables











//***** Global Variables and Functions *****/
Dim vrValue          as Variant
Dim sDataItems()     as String
Dim sOperatorType()  as String
Dim sActToUpdate()   as String

```

The following table explains the function of the editing/debugging icons and commands.

Icon	Key Combination	Function
	Ctrl+S	Save the current script to disk.
	Ctrl+O	Open an existing script.
	—	Create a new script.
	—	Save the current script to disk with new name.
	Ctrl+C	Copy the current editor selection to the Clipboard.

Icon	Key Combination	Function
	Shift+Del	Cut the current editor selection to the Clipboard.
	Ctrl+V	Paste the current contents of the Clipboard.
	Ctrl+Z	Undo the previous editor action.
	Shift+F2	Toggle the Console window.
	F2	Toggle the Variable window.
	F5	Execute the main procedure of the current Basic program if editing, or continue executing the current Basic program if debugging.
	Esc (twice) Ctrl+Break(twice)	Stop executing the current Basic program; if you are debugging the program, end the debugging session.
	Esc Ctrl+Break	Pause the currently executing Basic program and activate the debugger on the current line of execution.
	Ctrl+F5	Execute in Animate mode.
	F7	Sets a temporary breakpoint on the current line; runs the script up to the breakpoint, pauses, and clears the breakpoint.
	F9	Toggle a breakpoint on the current line.
	Ctrl+F8	Step out of the current procedure in the debugger.

Icon	Key Combination	Function
	F8	Step to the current line in the debugger, or begin debugging current file at main procedure.
	F10	Step over the current line in the debugger; if the current line is a call to a Basic subroutine or function, the debugger stops at the next line in the current procedure.
	F6	Check the syntax of the current Basic program by compiling the script.
	Alt+F3	Find and replace text.
	Ctrl+F	Find text in editor.
	F3 Shift+F3	Find next occurrence of last text searched for. Find previous occurrence of last text searched for.
	Ctrl+N	Scroll the Edit window to the next error returned from a syntax check.
	Ctrl+P	Scroll the Edit window to the previous error returned from a syntax check.
	F1	Display Help.
	—	Open dialog editor.

SureTrak Objects, Functions, and Properties

SureTrak contains objects that perform different actions, such as opening a project or calculating the critical path. By combining these actions with SBL programming techniques, you can save time by automating tasks you perform repeatedly. Objects control SureTrak as a whole (PrmApp object), the project window (PrmProjectWindow object), and activity-specific data (PrmProject object).

"Include" Files All flags, file IDs, and field IDs are defined in PRMCONST.SBH. Menu and toolbar commands are defined in PRMCMD.SBH. The SureTrak setup program copies these files to the SureTrak program directory when you install SureTrak. Type the following lines at the beginning of your SBL script to include the files:

```
'$include "prmconst.sbh"
'$include "prmcmd.sbh"
```

Sample SBL Script You can review any of the sample scripts delivered with SureTrak by choosing Tools, Basic Scripts. Select the script or executable you want, then click Edit. The code for the Cost Performance Report script follows; this script automatically runs a Cost Performance Report.

```
* Copyright 1997 Primavera Systems, Inc.      *
* COSTPERF.SBL - Calculates ETC based on CPI index and saves to project files *
'//***** SureTrak Header Files *****//
'$include "prmconst.sbh"
'$include "prmcmd.sbh"
'$include "wincmd.sbh"
'//*****//

Option Explicit      '// Flags undefined variables
Const SCRIPTNAME = "Cost Performance Index"
Const MSG_SAVE = "Save changes to project files"
Const MSG_COMPLETE = "Process successfully completed."
```

```

Sub Main()
    Dim prmwProj as New Prmprojectwindow
    Dim STW as New PrmApp
    Dim lOpen_proj as Long
    Dim sgEstimate_Cost as Single
    Dim sgActual_Cost as Single
    Dim sgEarned_Value as Single
    Dim sgBudget_Cost as Single
    Dim lRes_Pos as Long
    Dim sgPerf as Single
    lOpen_proj = prmwProj.BIND      '//Bind to Project displayed in Layout Window
    If (lOpen_proj = 0) then        '// No Projects Open
        If( prmwProj.Open = 0) then
            Exit Sub
        End If
    End if
    lRes_Pos = prmwProj().MOVEFIRST(FID_RES) '// Get First Resource in file; returns Postion
    '// in file
    While (lRes_Pos > 0)
        '// Retrieve Cost Information in Single Format
        sgEstimate_Cost = prmwProj().GET(FID_RES,RES_ETC)
        sgActual_Cost   = prmwProj().GET(FID_RES,RES_ATD)
        //////////////////////////////////////////
        lRes_Pos = prmwProj().MOVENEXT(FID_RES) '//Get Next Resource until EOF = 0
    Wend
    If MsgBox(MSG_SAVE, 36, SCRIPTNAME) = 6 Then prmwProj.Save      '// Save updated resources
    prmwProj.Refresh      '// Reflect changes
    lOpen_Proj = EnableWindow(STW.Hwnd, False)
    MsgBox MSG_COMPLETE, 64, SCRIPTNAME
    lOpen_Proj = EnableWindow(STW.Hwnd, True)
    lOpen_Proj = SetFocus(STW.Hwnd)
End Sub
End If

```

PrmApp Object The PrmApp functions (or methods) and properties control the application as a whole. Using PrmApp functions, you can exit SureTrak, log in entry rights data, return the product code, execute a script after a specific event occurs, and run a specified menu or toolbar command.

BackupFiles Function

Description: Creates a backup of a list of files.

Syntax: PrmApp.BackupFiles(list of files, destination filename, [flags])

Returns: Integer. FALSE (0) on error, otherwise TRUE (-1).

Remarks: Valid flags are in PRMCONST.SBH. *List of files* must be a binary file which contains a list of files to include in the backup. Each entry in the file must consist of two items. The first item is the zero-terminated filename of the file to include in the backup. Wildcards are permitted. The second entry is an integer value specifying whether an error should be produced if the file is not found (0 if no error is to be produced, 1 to produce an error).

BackupProject Function

Description: Creates a backup of a project.

Syntax: PrmApp.BackupProject([source project group name, [source member project name], destination project group name, [destination member project name], [type], [flags]])

Returns: Integer. FALSE (0) on error, otherwise TRUE (-1).

Remarks: If arguments are absent, Backup dialog is displayed. Valid types and flags are listed in PRMCONST.SBH.

Example: The following example uses the BackupProject function to backup the SureTrak project AUTO to ~AUTO.

```
app.BackupProject("C:\STWIN\PROJECTS\AUTO",
"C:\STWIN\PROJECTS\~AUTO")
```

Command Function

Description: Runs specified menu command.

Syntax: PrmApp.Command(command flag)

Returns: Integer. FALSE (0) on error, otherwise TRUE (-1).

Remarks: Valid command flags are listed in PRMCMD.SBH.

Example: The following example uses the Command function to start the SureTrak tutorial using the HELP_TUTORIAL command flag.app.
Command(HELP_TUTORIAL)

Delete Function

Description: Deletes the specified project.

Syntax: PrmApp.Delete([project group, [member project], [type]])

Returns: Integer. FALSE (0) on error, otherwise TRUE (-1).

Remarks: Opens the Delete dialog box if no parameters are specified.

DisableEvent Function

Description: Disables any event processing for the given event.

Syntax: PrmApp.DisableEvent(event)

Returns: Integer. FALSE (0) on error, otherwise TRUE (-1).

Remarks: Valid events are in listed PRMCONST.SBH. If event is less than 0, disables all events.

EnableEvent Function

Description: Enables any event processing for the given event.

Syntax: PrmApp.EnableEvent(event)

Returns: Integer. FALSE (0) on error, otherwise TRUE (-1).

Remarks: Valid events are listed in PRMCONST.SBH. If event is less than 0, enables all events.

EndEvent Function

Description: Removes the registration of a script.

Syntax: PrmApp.EndEvent(event, script, [subroutine])

Returns: Integer. FALSE (0) on error, otherwise TRUE (-1).

Remarks: Valid events are listed in PRMCONST.SBH.

Example: The following example uses the EndEvent function to unregister the foo script for the ON_CLOSE event.

```
app.EndEvent( ON_CLOSE, "C:\STWIN\SCRIPTS\F00.SBX")
```

Exit Function

Description: Exits SureTrak.

Syntax: PrmApp.Exit([flag])

Returns: Integer. FALSE (0) on error, otherwise TRUE (-1).

Remarks: Valid *flag* is SAVE_ON_EXIT. If no *flag* is specified, save will not be performed on exit.

GetEvent Function

Description: Returns the sequence number for a script and event.

Syntax: PrmApp.GetEvent(event, [script], [subroutine])

Returns: Integer. Value is 0 if an error occurred, otherwise the value is the sequence number for the event. If [script] and [subroutine] are not given, the number of events registered for this event is returned.

Remarks: Valid events are listed in PRMCONST.SBH.

Example: The following example uses the GetEvent function to test whether the foo script is already registered for the ON_CLOSE event by requesting its sequence number. If the foo script is not registered it will be registered using the OnEvent function.

```
sequence% = app.GetEvent( ON_CLOSE, "C:\STWIN\SCRIPTS\F00.SBX")
If( sequence = 0) Then
app.OnEvent(ON_CLOSE, "C:\STWIN\SCRIPTS\F00.SBX")
End If
```

GetTools Function

Description: Rereads the Tools menu from STWIN.INI.

Syntax: PrmApp.GetTools

Remarks: No arguments.

Hwnd Function

Description: Returns handle to window.

Syntax: PrmApp.Hwnd

Hwnd Function

Returns: Integer. Value is the handle to the application window.

Remarks: No arguments.

Example: The following example uses the Hwnd property to obtain the handle to the SureTrak Project Manager application window. It then uses the Microsoft Windows API function IsIconic to determine if the SureTrak application window has been minimized.

```
Declare Function IsIconic Lib "USER" ( ByVal hwnd as Integer) as Integer
handle% = app.Hwnd
minimized% = IsIconic( handle)
If( minimized = 1) Then
  MsgBox "SureTrak is minimized."
Else
  MsgBox "SureTrak is not minimized."
End If
```

Login Function

Description: Logs in entry rights data.

Syntax: PrmApp.Login(login, password, [type])

Returns: Integer. FALSE (0) on login failure, otherwise TRUE (-1).

Remarks: If no type is specified, default is PT_P3. Valid types are PT_P3 or PT_FH.

OnEvent Function

Description: Registers a script to be executed when a specified event occurs.

Syntax: PrmApp.OnEvent(event, script, [subroutine], [sequence number])

Returns: Integer. Value is 0 if error occurred, otherwise the value is the sequence number for the event.

Remarks: Valid events are listed in PRMCONST.SBH.

Example: The following example uses the OnEvent function to register the foo script if it is not already registered for the ON_CLOSE event. The GetEvent function is used to determine whether or not the script has already been registered.

```
sequence% = app.GetEvent(ON_CLOSE, "C:\STWIN\SCRIPTS\F00.SBX")
If( sequence = 0) Then
app.OnEvent( ON_CLOSE, "C:\STWIN\SCRIPTS\F00.SBX")
End If
```

Product Function

Description: Returns product code.

Syntax: PrmApp.Product

Returns: Long. Value is AT_SURETRAK20 (for version 2.0),
AT_SURETRAK15 (for version 1.5), or AT_SURETRAK (for version 1.0).

Remarks: No arguments.

RestoreFiles Function

Description: Restores a list of files.

Syntax: PrmApp.RestoreFiles(list of files, destination filename, [flags])

Returns: Integer. FALSE (0) on error, otherwise TRUE (-1).

Remarks: Valid flags are listed in PRMCONST.SBH. *List of files* must be a binary file which contains a list of files to restore. Each entry in the file must consist of two items. The first item is the zero-terminated filename for the file to be restored. Wildcards are permitted. The second entry is an integer value specifying whether an error should be produced if the file is not found (0 if no error is to be produced, 1 to produce an error).

RestoreProject Function

Description: Restores a project.

Syntax: PrmApp.RestoreProject([source name [fsub], destination name,[tsub], [type],
[flags]])

Returns: Integer. FALSE (0) on error, otherwise TRUE (-1).

Remarks: Valid types and flags are listed in PRMCONST.SBH.

Example: The following example uses the RestoreProject function to restore the ~AUTO SureTrak project to AUTO.

```
app.RestoreProject( "C:\STWIN\PROJECTS\~AUTO", "C:\STWIN\PROJECTS\AUTO")
```

RunEvent Function

Description: Executes all the scripts registered with an event.

Syntax: PrmApp.RunEvent(event, [arguments])

Returns: Integer. FALSE (0) on error, otherwise TRUE (-1).

Remarks: Valid events are listed in PRMCONST.SBH.

Example: The following example uses the RunEvent function to run all scripts associated with the ON_CLOSE event.app.
RunEvent(ON_CLOSE)

SendMail Function

Description: Displays the Mail dialog box, allowing the user to send messages.

Syntax: PrmApp.SendMail

Returns: Integer. FALSE (0) on error, otherwise TRUE (-1).

Remarks: No arguments.

PrmProject Window Object The PrmProjectWindow functions (or methods) and properties control the project window. Using PrmProjectWindow, you can open a project, create a new project, implement a layout, implement a filter, save a project, close a project, repaint the project window, print a report, run a specified menu or toolbar command, execute a script after a specific event occurs, set and get the hidden or selection status of a current activity, and return the project tied to the project window.

Bind Function

Description: Binds PrmProjectWindow to a currently open project window.

Syntax: PrmProjectWindow.Bind([project group name, [member project name]])

Returns: Integer. FALSE (0) on error, otherwise TRUE (-1).

Remarks: If *project group name* is absent, binds to active (top) project.

Example: The following example uses the Bind function to bind a PrmProjectWindow object to the C:\STWIN\PROJECTS\AUTO project. If the project is not open then it will open the project using the Open function.

```
bind% = projwin.Bind( "C:\STWIN\PROJECTS\AUTO")
If( bind = 0) Then
bind = projwin.Open( "C:\STWIN\PROJECTS\AUTO")
End If
```

Close Function

Description: Closes the project window without saving.

Syntax: PrmProjectWindow.Close

Returns: Integer. FALSE (0) on error, otherwise TRUE (-1).

Remarks: No arguments.

Command Function

Description: Runs specified menu command.

Syntax: PrmProjectWindow.Command(command flag)

Returns: Integer. FALSE (0) on error, otherwise TRUE (-1).

Remarks: Valid command flags are listed in PRMCMD.SBH.

Example: The following example selects the first four activities in the FID_ACT file using the MoveFirst, MoveNext, and Select functions. It then uses the Command function to copy the activity information to the Clipboard.

```
projwin.project.MoveFirst( FID_ACT)
projwin.Select = TRUE
For i = 1 To 3
projwin.project.MoveNext( FID_ACT)
projwin.Select = TRUE
Next
projwin.Command( EDIT_COPY)
```

Create Function

Description: Creates a new project.

Syntax: PrmProjectWindow.Create([project group name, [member project name], [project ID], [template project], [type]])

Returns: Integer. FALSE (0) on error, otherwise TRUE (-1).

Remarks: If *project group name* is absent, displays New Project dialog box. If *project ID* is absent, uses first two characters of member project name. For example, a member project named ENGR uses EN as its project ID if none was explicitly specified. The template project is an existing project to use when creating the new project, and must be in the TEMPLATE directory.

Example: The following example creates a CLNP member project in the CNCT P3 project group.

```
projwin.Create( "C:\STWIN\PROJECTS\CNCT", "CLNP")
```

EndEvent Function

Description: Removes the registration of a script.

Syntax: PrmProjectWindow.EndEvent(event, script, [subroutine])

Returns: Integer. FALSE (0) on error, otherwise TRUE (-1).

Remarks: Valid events are listed in PRMCONST.SBH.

Example: The following example uses the EndEvent function to unregister the foo script for the ON_PROJECT_CLOSE event.

```
projwin.EndEvent( ON_PROJECT_CLOSE, "C:\STWIN\SCRIPTS\F00.SBX")
```

Filter Function

Description: Applies a specified filter.

Syntax: PrmProjectWindow.Filter([filter name, [mode], [action]])

Returns: Integer. FALSE (0) on error, otherwise TRUE (-1).

Remarks: If *filter name* is not specified, the Filter dialog box opens. Valid modes are

FILTER_REPLACE
FILTER_APPEND
FILTER_REMOVE

Valid actions are

FILTER_HIDE
FILTER_SELECT

If *mode* is not specified, mode is FILTER_REPLACE.

If *action* is not specified, action is FILTER_HIDE.

Example: The following example uses the Filter function to replace any currently applied filter with the CRIT filter, selecting the activities specified by the filter.

```
projwin.Filter("CRIT", FILTER_REPLACE, FILTER_SELECT)
```

GetEvent Function

Description: Returns the sequence number for a script and event.

Syntax: PrmProjectWindow.GetEvent(event, [script], [subroutine]))

Returns: Integer. Value is 0 if an error occurred; otherwise, returns the sequence number for the event. If *script* and *subroutine* are not given, returns the number of events registered.

Remarks: Valid events are listed in PRMCONST.SBH.

Example: The following example uses the GetEvent function to test if the foo script is already registered for the ON_PROJECT_CLOSE event by requesting its sequence number. If the foo script is not registered, it will be registered using the OnEvent function.

```
seq% = projwin.GetEvent( ON_PROJECT_CLOSE, "C:\STWIN\SCRIPTS\F00.SBX")
If( seq = 0) Then
projwin.OnEvent( ON_PROJECT_CLOSE, "C:\STWIN\SCRIPTS\F00.SBX")
End If
```

Hide Property

Description: Sets or gets hidden status of current activity.

Syntax: PrmProjectWindow.Hide

Returns: Integer. Value is TRUE for hidden, FALSE for visible.

Remarks: Valid values are TRUE/FALSE.

Example: The following example uses the MoveFirst and Get functions, and the Project and Hide properties, to hide all hammock activities.

```
mark% = projwin.Project.MoveFirst( FID_ACT)
While( mark <> 0)
type% = projwin.Project.Get( FID_ACT, ACT_TYPE)
If( type = ACT_TYPE_HAMMOCK) Then
projwin.Hide = TRUE
End If
mark = projwin.Project.MoveNext( FID_ACT)
Wend
```

Hwnd Function

Description: Returns handle to window.

Syntax: PrmProjectWindow.Hwnd

Returns: Integer. Value is the handle to the application window.

Remarks: No arguments.

Example: The following example uses the Hwnd properties for both the PrmProjectWindow and the PrmApp to maximize the project window.

```
Declare Function SendMessage Lib "USER" ( ByVal hwnd as Integer, ByVal msg as Integer, ByVal wParam as Integer, ByVal lParam as Long) as Long const  
WM_MDIMAXIMIZE = &H0225  
appHandle% = app.Hwnd  
projwinHandle% = projwin.Hwnd  
SendMessage( appHandle, WM_MDIMAXIMIZE, projwinHandle, 0);
```

Layout Subroutine

Description: Applies the specified layout.

Syntax: PrmProjectWindow.Layout [layout name] [dontpromptsave]

Returns: Not applicable.

Remarks: If *layout name* is not specified, the Layout Control dialog box opens. If the current layout has been modified, SureTrak prompts to save the changes unless *dontpromptsave* is set to TRUE.

Example: The following example uses the Layout function to apply the WBS layout.

```
PrmProjectWindow.Layout( "WBS")
```

OnEvent Function

Description: Registers a script to be executed when a specified event occurs.

Syntax: PrmProjectWindow.OnEvent(event, script, [subroutine], [sequence number])

Returns: Integer. Value is 0 if an error occurred; otherwise, the value is the sequence number for the event.

Remarks: Valid events are listed in PRMCONST.SBH.

Example: The following example uses the OnEvent function to register the foo script if it is not already registered for the ON_PROJECT_CLOSE event. The GetEvent function is used to determine whether or not the script has already been registered.

```
seq% = projwin.GetEvent( ON_PROJECT_CLOSE, "C:\STWIN\SCRIPTS\F00.SBX")
If( seq = 0) Then
projwin.OnEvent( ON_PROJECT_CLOSE, "C:\STWIN\SCRIPTS\F00.SBX")
End If
```

Open Function

Description: Opens the specified project.

Syntax: PrmProjectWindow.Open([project group name, [member project name], [type]])

Returns: Integer. FALSE (0) on error, otherwise TRUE (-1).

Remarks: If a *project group name* is not specified, the Open Project dialog box is opened. Valid type values are

```
PT_SURETRAK
PT_P3
PT_FH
PT_MPX
PT_ST20
PT_GROUPS
```

Example: The following example uses the Open function to open the ENGR member project in the CNCT P3 project group.

```
projwin.Open( "C:\STWIN\PROJECTS\CNCT", "ENGR", PT_P3)
```


Print Function

Description: Prints the specified report.

Syntax: PrmProjectWindow.Print([report name],[flag])

Returns: Integer. FALSE (0) on error, otherwise TRUE (-1).

Remarks: If *flag* is not specified, opens the Print dialog box. If *report name* is not specified, prints current project window. Valid flag is PRINT_NOSHOW_DLG.

Example: The following example uses the Print function to print the GNT1 report without displaying the print dialog box.

```
projwin.Print("GNT1", PRINT_NOSHOW_DLG)
```

Project Property

Description: Returns the PrmProject tied to this PrmProjectWindow.

Syntax A: PrmProjectWindow.Project.Function

Syntax B: PrmProjectWindow().Function

Returns: PrmProject Object.

Remarks: Valid functions are any PrmProject functions.

Example: The following example uses the Project property to run the Get function in the PrmProject Object, retrieving the Activity ID for the current activity.

```
actnum% = projwin.Project.Get( FID_ACT, ACT_ACTID)
```

Refresh Function

Description: Repaints the project window.

Syntax: PrmProjectWindow.Refresh

Returns: Integer. FALSE (0) on error, otherwise TRUE (-1).

Remarks: No arguments.

Reset Function

Description: Closes the project window with save.

Syntax: PrmProjectWindow.Reset(Dlg)

Returns: Integer. FALSE (0) on error, otherwise TRUE (-1).

Remarks: If *Dlg* is TRUE, displays save confirmation message; if FALSE, saves project without confirmation.

RunEvent Function

Description: Executes all the scripts registered with an event.

Syntax: PrmProjectWindow.RunEvent(event)

Returns: Integer. FALSE (0) on error, otherwise TRUE (-1).

Remarks: Valid events are listed in PRMCONST.SBH.

Example: The following example uses the RunEvent function to run all scripts associated with the ON_PROJECT_CLOSE event.

```
projwin.RunEvent( ON_PROJECT_CLOSE)
```

Save Function

Description: Saves the project window, but does not close it.

Syntax: PrmProjectWindow.Save

Returns: Integer. FALSE (0) on error, otherwise TRUE (-1).

Remarks: No arguments.

SaveAs Function

Description: Saves the project to the specified name.

Syntax: PrmProjectWindow.SaveAs([project group name],[member projectname],[project ID],
[type], [don't reopen]))

Returns: Integer. FALSE (0) on error, otherwise TRUE (-1).

Remarks: If *project group name* is not specified, displays Save As dialog box. Valid types are

PT_SURETRAK
PT_P3
PT_FH
PT_PGROUPTS
PT_MPX

The *project group name* is the name of the project group in which the member project should be created if *member project name* is also specified; otherwise, *project group name* is the name with which to save the current standalone project. The *project ID* is a unique two-character ID that SureTrak prepends to all activities in the member project. If *project ID* is not specified, the first two characters of the member project name are used; if the *project ID* already exists (that is, it is not unique among projects in the current directory), an error occurs. The *don't reopen* flag applies only when saving as a member project. If this parameter is set to True (-1), the member project will not be reloaded after it is created; a value of False (0) reloads the member project.

Example: The following example uses the SaveAs function to save the current project as a P3 type project called PROJ.

```
projwin.SaveAs("PROJ", , , PT_P3)
```

Select Property

Description: Sets or gets the selection status of the current activity.

Syntax: PrmProjectWindow.Select

Returns: Integer. Value is TRUE (-1) for a selected activity, FALSE (0) for unselected.

Remarks: Valid values are TRUE and FALSE.

Example: The following example selects the first four activities in the FID_ACT file using the MoveFirst, MoveNext, and Select functions. It then uses the Command function to copy the activity information to the Clipboard.

```
projwin.project.MoveFirst( FID_ACT)
projwin.Select = TRUE
For i = 1 To 3
projwin.project.MoveNext( FID_ACT)
projwin.Select = TRUE
Next
projwin.Command( EDIT_COPY)
```

SendMail Function

Description: Opens a blank mail message window.

Syntax: PrmProjectWindow.SendMail

Returns: Integer. FALSE (0) on error, otherwise TRUE (-1).

Remarks: Displays the Mail dialog box and allows the user to send a message. Cannot be used to send project data.

PrmProject Object The PrmProject functions (or methods) and properties control the project data. Using PrmProject, you can access project data, save project data, close the project, return the project group name, return the number of items in a file, add and move among bookmarks within project data, delete data, recalculate the schedule, level resources, update as of a specified data date, and rebuild a specified file.

AddNew Function

Description: Creates a new record in the specified file.

Syntax: PrmProject.AddNew(FID, key1,[key2])

Returns: Long. Value is the current item number for the specified file, 0 on error.

Remarks: **Key 1** indicates the record to add. Valid file IDs are

FID_ACT, where **key1** is the Activity ID.

FID_PRED, where **key 1** is the predecessor Activity ID and **key2** is the successor Activity ID.

FID_SUCC, where **key1** is the successor Activity ID and **key2** is the predecessor Activity ID. FID_RLB, where **key1** is the resource ID.

FID_RES, where **key1** is the resource ID and **key2** is the Activity ID to which the resource is to be tied.

FID_TTL and FID_WBS, where **key1** is the activity code value, Activity ID, WBS code, or WBS structure to be added, and **key2** is the code type to which the value is to be added.

AddNew Function

Example: The following example uses the AddNew and Put functions to add three new activities, each with a duration of 80 hours, and create finish to start relationships between them.

```
proj.AddNew( FID_ACT, "A100")
proj.put( FID_ACT, ACT_ORGDUR) = 80
proj.AddNew( FID_ACT, "A110")
proj.Put( FID_ACT, ACT_ORGDUR) = 80
proj.AddNew( FID_ACT, "A120")
proj.Put( FID_ACT, ACT_ORGDUR) = 80
proj.AddNew( FID_PRED, "A100", "A110")
proj.Put( FID_SUCC, REL_TYPE) = REL_TYPE_FS
proj.AddNew( FID_PRED, "A110", "A120")
proj.Put( FID_SUCC, REL_TYPE) = REL_TYPE_FS
```

The following example uses the AddNew function to add the resource ENGNR to the resource dictionary and assign the ENGNR resource to activity A100.

```
proj.AddNew( FID_RLB, "ENGNR")
proj.AddNew( FID_RES, "ENGNR", "A100")
```

The following example uses the AddNew function to add the activity code RESP as the first activity code, and to add the value PETE as an RESP code value.

```
proj.AddNew( FID_TTL, "RESP", "A")
proj.AddNew( FID_TTL, "PETE", "a")
```

The following example uses the AddNew function to define a WBS structure of three levels, each with a length of one character, and to add the WBS code 1.2.1 to the WBS dictionary.

```
struct$ = chr$( 1) + "." + chr$( 1) + "." + chr$( 1)
code$ = "1.2.1"
proj.AddNew( FID_WBS, struct, "y")
proj.AddNew( FID_WBS, code)
```

BackupProject Function

Description: Creates a backup of the current project.

Syntax: PrmProject.BackupProject(destination filename, [sub], [flags])

Returns: Integer. FALSE (0) on error, otherwise TRUE (-1).

Remarks: Specify the project ID of a member project in *sub* if backing up a member project within a project group. Valid *flags* values are located in PRMCONST.SBH.

Example: The following example uses the BackupProject function to back up the current SureTrak-type project to C:\STWIN\BACKUP.

```
proj.BackupProject("C:\STWIN\BACKUP)
```

Bind Function

Description: Binds PrmProject to a currently open project window.

Syntax: PrmProject.Bind([project group name, [member project name]])

Returns: Integer. FALSE (0) on error, otherwise TRUE (-1).

Remarks: If the project group name is not specified, binds to active (top) project.

Example: The following example uses the Bind function to bind the PrmProject object to the C:\STWIN\PROJECTS\AUTO project data. If the project is not open, it will open the project using the Open function.

```
bind% = proj.Bind("C:\STWIN\PROJECTS\AUTO")
If( bind = 0) Then
proj.Open( "C:\STWIN\PROJECTS\AUTO")
End If
```

Bookmark Property

Description: A bookmark is a handle to a specific item in the specified file.

Syntax: PrmProject.Bookmark(FID)

Returns: Long. Value will be the item number for the specified file, or 0 on error.

Remarks: Valid file IDs are

- FID_ACT
- FID_PRED FID_SUCC
- FID_RLB
- FID_RES
- FID_TTL
- FID_WBS

Example: The following example uses the Bookmark property to save the position of the current activity while using the FindFirst function to determine whether activity A100 exists.

```
position% = proj.Bookmark
exists% = FindFirst( FID_ACT, "A100")
proj.Bookmark = position
```

Close Function

Description: Closes the current project without saving.

Syntax: PrmProject.Close

Returns: Integer. FALSE (0) on error, otherwise TRUE (-1).

Remarks: No arguments.

Count Property

Description: Returns the number of items in the specified file.

Syntax: PrmProject.Count(FID)

Returns: Long. Value is the number of items in the file.

Remarks: Valid file IDs are

FID_ACT
FID_PRED
FID_SUCC
FID_RLB
FID_RES
FID_TTL
FID_WBS

Example: The following example uses the Count property to get the number of records in the FID_ACT file.

```
numacts% = proj.Count( FID_ACT)
msg$ = "There are " + Str$( numacts) + " activities."
MsgBox msg
```

Create Function

Description: Creates a new project.

Syntax: PrmProject.Create([project group name, [member project name], [project ID], [template project], [type]])

Returns: Integer. FALSE (0) on error, otherwise TRUE (-1).

Remarks: If *project group name* is not specified, SureTrak displays the New Project dialog box. If *project ID* is not specified, SureTrak uses the first two characters of the *member project name*. For example, a member project named ENGR uses EN as its project ID, if none was specified. The template project is an existing project to use when creating the new project.

Example: The following example creates a CLNP member project in the CNCT P3 project group.

```
proj.Create( "C:\STWIN\PROJECTS\CNCT", "CLNP")
```

Delete Function

Description: Deletes an item in the specified file.

Syntax: PrmProject.Delete(FID, [bookmark])

Returns: Integer. FALSE (0) on error, otherwise TRUE (-1).

Remarks: If a bookmark is absent, the current item of the specified file is deleted. Valid file IDs are

FID_ACT
FID_PRED
FID_SUCC
FID_RLB
FID_RES
FID_TTL
FID_WBS

Example: The following example uses the Delete function to delete the current resource assignment.
`proj.Delete(FID_RES)`

EndEvent Function

Description: Removes the registration of a script.

Syntax: PrmProject.EndEvent(event, script, [subroutine])

Returns: Integer. FALSE (0) on error, otherwise TRUE (-1).

Remarks: Valid events are listed in PRMCONST.SBH.

Example: The following example uses the EndEvent function to unregister the foo script for the ON_PROJECT_CLOSE event.
`proj.EndEvent(ON_PROJECT_CLOSE, "C:\STWIN\SCRIPTS\F00.SBX")`



When you use *FindFirst*, *FindLast*, *FindNext*, or *FindPrevious* with activity, Activity ID, and WBS codes, you must identify the type of value you are working with. Activity code fields are identified by an uppercase letter; the first code is A, the next is B, and so on, through T. The four Activity ID code fields are identified by the letters U through X. The WBS structure is identified with a lowercase y; and the WBS code value list is identified with a lowercase z. Prepend the "dictionary ID" (the letters specified above) to the code value. For example, to work with the value DAVE in the Responsibility (RESP) code field, assuming that RESP is the first activity code field, specify ADAVE.

FindFirst Function

Description: Finds first item equal to or between parameter(s).

Syntax: PrmProject.FindFirst(FID, key1,[key2])

Returns: Long. Value is the current item number for the specified file, or 0 on error.

Remarks: Returns a bookmark. Valid file IDs (*FID*) are

FID_ACT
FID_PRED
FID_SUCC
FID_RLB
FID_RES
FID_TTL
FID_WBS

Example: The following example uses the FindFirst and FindNext functions to find all activities with Activity IDs between A100 and A200.

```
activity% = proj.FindFirst( FID_ACT, "A100", "A200")
While( activity <> 0) Then
activity = proj.FindNext( FID_ACT, "A100", "A200")
Wend
```

FindLast Function

Description: Finds last item equal to or between parameter(s).

Syntax: PrmProject.FindLast(FID, key1,[key2])

Returns: Long. Value is the current item number for the specified file, or 0 on error.

Remarks: Returns a bookmark. Valid file IDs (*FID*) are

FID_ACT
FID_PRED
FID_SUCC
FID_RLB
FID_RES
FID_TTL
FID_WBS

Example: The following example uses the FindLast and FindPrevious functions to find all activities with Activity IDs between A100 and A200.

```
activity% = proj.FindLast( FID_ACT, "A100", "A200")  
While( activity <> 0) Then  
activity = proj.FindPrevious( FID_ACT, "A100", "A200")  
Wend
```

FindNext Function

Description: Finds next item equal to or between parameter(s).

Syntax: PrmProject.FindNext(FID, key1,[key2])

Returns: Long. Value is the current item number for the specified file, or 0 on error.

Remarks: Returns a bookmark. Valid file IDs (*FID*) are

FID_ACT
FID_PRED
FID_SUCC
FID_RLB
FID_RES
FID_TTL
FID_WBS

Example: The following example uses the FindFirst and FindNext functions to find all activities with Activity IDs between A100 and A200.

```
activity% = proj.FindFirst( FID_ACT, "A100", "A200")  
While( activity <> 0) Then  
activity = proj.FindNext( FID_ACT, "A100", "A200")  
Wend
```

FindPrevious Function

Description: Finds previous item equal to or between parameter(s).

Syntax: PrmProject.FindPrevious(FID, key1,[key2])

Returns: Long. Value is the current item number for the specified file, or 0 on error.

Remarks: Returns a bookmark. Valid file IDs (*FID*) are

FID_ACT
FID_PRED
FID_SUCC
FID_RLB
FID_RES
FID_TTL
FID_WBS

Example: The following example uses the FindLast and FindPrevious functions to find all activities with Activity IDs between A100 and A200.

```
activity% = proj.FindLast( FID_ACT, "A100", "A200")
While( activity <> 0) Then
  activity = proj.FindPrevious( FID_ACT, "A100", "A200")
Wend
```

Flag Property

Description: Sets or gets status of a specified project flag.

Syntax: PrmProject.Flag(flag ID)

Returns: Integer. Value is TRUE if mode is on, FALSE if mode is off.

Remarks: Valid *flag IDs* are

PROJECT_FLAG_SCHEDULE
PROJECT_FLAG_LEVEL
PROJECT_AUTOORGANIZE

Example: The following example uses the Flag property to turn automatic reorganization off.

```
proj.Flag( PROJECT_FLAG_AUTOORGANIZE) = FALSE
```



This property is identical to the ScheduleMode and LevelMode properties.

GetEvent Function

Description: Returns the sequence number for a script and event.

Syntax: PrmProject.GetEvent(event, [script], [subroutine])

Returns: Integer. Value is 0 if an error occurred; otherwise the value is the sequence number for the event. If *script* and *subroutine* are not specified, returns the number of events registered.

Remarks: Valid events are listed in PRMCONST.SBH.

Example: The following example uses the GetEvent function to test if the foo script is already registered for the ON_PROJECT_CLOSE event by requesting its sequence number. If the foo script is not registered, it will be registered using the OnEvent function.

```
seq% = proj.GetEvent( ON_PROJECT_CLOSE, "C:\STWIN\SCRIPTS\F00.SBX")
If( seq = 0) Then
proj.OnEvent( ON_PROJECT_CLOSE, "C:\STWIN\SCRIPTS\F00.SBX")
End If
```

Get Function

Description: Get field data in the specified file.

Syntax A: PrmProject.Get(FID, field ID)

Syntax B: PrmProject(FID, field ID)

Returns: Variant.

Remarks: File and field IDs are listed in PRMCONST.SBH.

Level Function

Description: Immediate resource leveling.

Syntax: PrmProject.Level

Returns: Integer. FALSE (0) on error, otherwise TRUE (-1).

Remarks: No arguments.

LevelMode Property

Description: Turns automatic leveling on or off.

Syntax: PrmProject.LevelMode

Returns: Integer. Value is MODE_ON if automatic leveling is on, MODE_OFF if automatic leveling is off.

MoveFirst Function

Description: Moves to the first item in the file.

Syntax: PrmProject.MoveFirst(FID)

Returns: Long. Value is the current item number for the specified file, or 0 on error.

Remarks: Valid file IDs (*FID*) are

FID_ACT
FID_PRED
FID_SUCC
FID_RLB
FID_RES
FID_TTL
FID_WBS

Example: The following example uses the MoveFirst, MoveNext, Get, and Put functions to increase all activity durations by 8 hours.

```
activity% = proj.MoveFirst( FID_ACT)
While( activity <> 0) Then
dur% = proj.Get( FID_ACT, ACT_ORGDUR)
dur = dur + 8
proj.Put( FID_ACT, ACT_ORGDUR) = dur
activity = proj.MoveNext( FID_ACT)
Wend
```

MoveLast Function

Description: Moves to the last item in the file.

Syntax: PrmProject.MoveLast(FID)

Returns: Long. Value is the current item number for the specified file, or 0 on error.

Remarks: Valid file IDs (*FID*) are

FID_ACT
FID_PRED
FID_SUCC
FID_RLB
FID_RES
FID_TTL
FID_WBS

MoveNext Function

Description: Moves to the next item in the file.

Syntax: PrmProject.MoveNext(FID)

Returns: Long. Value is the current item number for the specified file, or 0 on error.

Remarks: Valid file IDs (*FID*) are

FID_ACT
FID_PRED
FID_SUCC
FID_RLB
FID_RES
FID_TTL
FID_WBS

Example: The following example uses the MoveFirst, MoveNext, Get, and Put functions to increase all activity durations by 8 hours.

```
activity% = proj.MoveFirst( FID_ACT)
While(activity <> 0) Then
dur% = proj.Get( FID_ACT, ACT_ORGDUR)
dur = dur + 8
proj.Put( FID_ACT, ACT_ORGDUR) = dur
activity = proj.MoveNext( FID_ACT)
Wend
```

MovePrevious Function

Description: Moves to the previous item in the file.

Syntax: PrmProject.MovePrevious(FID)

Returns: Long. Value is the current item number for the specified file, or 0 on error.

Remarks: Valid file IDs (*FID*) are

FID_ACT
FID_PRED
FID_SUCC
FID_RLB
FID_RES
FID_TTL
FID_WBS

Example: The following example uses the MoveLast, MovePrevious, Get, and Put functions to increase all activity durations by 8 hours.

```
activity% = proj.MoveLast( FID_ACT)
While(activity <> 0) Then
dur% = proj.Get( FID_ACT, ACT_ORGDUR)
dur = dur + 8
proj.Put( FID_ACT, ACT_ORGDUR) = dur
activity = proj.MovePrevious( FID_ACT)
Wend
```

Name Function

Description: Returns the project group (or standalone project) name.

Syntax: PrmProject.Name

Returns: Variant. Variant type is String, and contains the name of the project group.

Remarks: No arguments.

OnEvent Function

Description: Registers a script to be executed when a specified event occurs.

Syntax: PrmProject.OnEvent(event, script, [subroutine], [sequence number])

Returns: Integer. Value is 0 if error occurred; otherwise, the value is the sequence number for the event.

Remarks: Valid events are listed in PRMCONST.SBH.

Example: The following example uses the OnEvent function to register the foo script if it is not already registered for the ON_PROJECT_CLOSE event. The GetEvent function is used to determine whether or not the script has already been registered.

```
seq% = proj.GetEvent(ON_PROJECT_CLOSE, "C:\STWIN\SCRIPTS\F00.SBX")
If(seq= 0) Then
proj.OnEvent(ON_PROJECT_CLOSE, "C:\STWIN\SCRIPTS\F00.SBX")
EndIf
```

Open Function

Description: Opens the project.

Syntax: PrmProject.Open([project group name],[member project name],[type])

Returns: Integer. FALSE (0) on error, otherwise TRUE (-1).

Remarks: If *project group name* is absent, SureTrak displays the Open Project dialog box. Valid *types* are

```
PT_SURETRAK
PT_P3PT_FH
PT_MPX
PT_ST20
PT_PGROUPTS
```

Example: The following example uses the Open function to open the ENGR member project in the CNCT P3 project group.

```
proj.Open("C:\STWIN\PROJECTS\CNCT", "ENGR", PT_P3)
```

Profile Function

Description: Displays a specified Resource profile.

Syntax: PrmProject.Profile(date, array, [scale], [type], [activity], [resource])

Returns: Double; FALSE if date is 0.

Remarks: *date* is the beginning date for the profile

array specifies the number of days, weeks, or months of data to store and report in the profile

scale sets the timescale of the profile (see Scale Flags in PRMCONST.SBH)

type can be units, costs, revenue, or others (see Type Flags in PRMCONST.SBH)

activity restricts the profile to a specified activity; if blank, the profile includes all activities

resource restricts the profile to a specified resource; if blank, the profile includes all resources

Example: The following example retrieves three weeks' costs for activity 1000 beginning 09Jun97.

```
Dim Cost(3) as Double
Dim Week1 as Double
Week1 = Proj.Profile( "09JUN97", Cost, Profile_Week, Profile_Costs,
"1000", " ")
```

Put Function

Description: Puts field data in the specified file.

Syntax A: PrmProject.Put(FID, field ID)

Syntax B: PrmProject(FID, field ID)

Example: The following example uses the Put function to set the duration for the current activity to 80 hours.

```
proj.Put(FID_ACT, ACT_ORGDUR) = 80
```

RepairDatabase Function

Description: Rebuilds the index of the specified file.

Syntax: PrmProject.RepairDatabase[FID])

Returns: Integer. FALSE (0) on error, otherwise TRUE (-1).

Remarks: Valid file IDs (*FID*) are

FID_ACT
FID_PRED
FID_SUCC
FID_RLB
FID_RES
FID_TTL
FID_WBS

Reset Function

Description: Closes the project with save.

Syntax: PrmProject.Reset

Returns: Integer. FALSE (0) on error, otherwise TRUE (-1).

Remarks: No arguments.

RunEvent Function

Description: Executes all the scripts registered with an event.

Syntax: PrmProject.RunEvent(event, [arguments])

Returns: Integer. FALSE (0) on error, otherwise TRUE (-1).

Remarks: Valid *events* are listed in PRMCONST.SBH.

Example: The following example uses the RunEvent function to run all scripts associated with the ON_PROJECT_CLOSE event.

```
proj.RunEvent( ON_PROJECT_CLOSE)
```

Save Function

Description: Saves the project without closing.

Syntax: PrmProject.Save

Returns: Integer. FALSE (0) on error, otherwise TRUE (-1).

SaveAs Function

Description: Saves the project to the specified name.

Syntax: PrmProject.SaveAs([project group name],[member projectname],[project ID],[type],
[don't reopen]])

Returns: Integer. FALSE (0) on error, otherwise TRUE (-1).

Remarks: If *project group name* is not specified, displays Save As dialog box. Valid types are

PT_SURETRAK
PT_P3
PT_FH
PT_PGROUPTS
PT_MPX

The *project group name* is the name of the project group in which the member project should be created if *member project name* is also specified; otherwise, *project group name* is the name with which to save the current standalone project. The *project ID* is a unique two-character ID that SureTrak prepends to all activities in the member project. If *project ID* is not specified, the first two characters of the member project name are used; if the *project ID* already exists (that is, it is not unique among projects in the current directory), an error occurs. The *don't reopen* flag applies only when saving as a member project. If this parameter is set to True (-1), the member project will not be reloaded after it is created; a value of False (0) reloads the member project.

Example: The following example uses the SaveAs function to save the current project as a P3 type project called PROJ.

```
PrmProject.SaveAs("PROJ", , , PT_P3)
```


Schedule Function

Description: Immediate schedule recalculation.

Syntax: PrmProject.Schedule

Returns: Integer. FALSE (0) on error, otherwise TRUE (-1).

Remarks: No arguments.

ScheduleMode Property

Description: Turns automatic scheduling on or off.

Syntax: PrmProject.ScheduleMode

Returns: Integer. Value is MODE_ON (-1) if automatic scheduling is on, MODE_OFF (0) if automatic scheduling is off.

SendMail Function

Description: Sends a mail message or project information as part of a mail message.

Syntax: PrmProject.SendMail(arguments)

Returns: Integer. FALSE (0) on error, otherwise TRUE (-1).

SendMail Function

Remarks: *Arguments* specify the information to send as an attachment to the mail message. Specify one or more arguments, separating them with a "+". (Note: do not combine MAIL_MESSAGE with any other argument.) Arguments are listed in PRMCONST.SBH:

MAIL_PROJECT sends a compressed project backup

MAIL_SIMPLEBMP sends a 16-color bitmap of the layout

MAIL_DETAILBMP sends a bitmap of the layout at your system's current resolution

MAIL_ALLUPD sends standard update information for all activities

MAIL_SELECTUPD sends standard update information for currently selected activities

MAIL_CLIP sends the contents of the Clipboard

MAIL_CODES generates mail messages to all E-mail addresses listed in the activity code field identified for MAIL classification

MAIL_MESSAGE sends a text message

Type Function

Description: Specifies the type of project open.

Syntax: PrmProject.type

Returns: Project type flag.

Remarks: Type flags are

PT_SURETRAK

PT_P3

PT_FH

PT_MPX

PT_ST20

PT_PGROUPTS

Update Function

Description: Immediate updating as of specified data date.

Syntax: PrmProject.Update([data date])

Returns: Integer. FALSE (0) on error, otherwise TRUE (-1).

Remarks: If *data date* is not specified, SureTrak displays the Project Status dialog box.

Example: The following example (which assumes that SureTrak is configured to use the mmdddy date format) uses the Update function to update the project as of September 1, 1997.

```
Dim datadate as Variant  
datadate = 090197  
proj.Update(datadate)
```

Progress Meter Functions You can display a progress meter onscreen to provide a visual indication of the progress of lengthy functions. For example, indicate progress while globally changing data or running a series of reports. Progress meter functions are declared for you in the WINCMD.SBH file. You can include this file and access these functions in any SBL script.

StartProgress Function

Description: Starts the progress meter, or resets it to zero if it is already running. Specify title and content text for the meter.

Syntax: StartProgress(szTitle, szText, iMin, iMax)

Returns: Integer. FALSE (0) on error, otherwise TRUE (-1).

Remarks: *szTitle* is the title bar text for the status meter.

SzText is the text to be displayed in the status meter.

iMin is the minimum value the meter can display.

iMax is the maximum value the meter can display.

Execution of this function automatically starts STSTATUS.EXE, which continues to run until the execution of the EndProgress function.

Only one instance of the status meter can run at any time.

Example: iRVal = StartProgress("SureTrak Project Manager", "Processing...", 0, 100)

UpdateProgress Function

Description: Updates the progress meter to the value of iPos.

Syntax: UpdateProgress(iPos)

Returns: Integer. FALSE (0) on error, otherwise TRUE (-1).

Remarks: *iPos* is the position to which to update the status meter. The execution of this function requires prior execution of the StartProgress function. Values smaller than the *iMin* or larger than the *iMax* specified with StartProgress result in an error.

Example: This example, which assumes the range of the previous example, updates the progress meter to 50 percent.

```
iRVal =UpdateProgress( 50)
```

EndProgress Function

Description: Closes the progress meter and exits STSTATUS.EXE.

Syntax: EndProgress()

Returns: Integer. FALSE (0) on error, otherwise TRUE (-1).

Remarks: No parameters. Use this function before the conclusion of the function whose progress is being measured. Use EndProgress to close the status meter when you will not be using it again immediately.

Example: This example closes the progress meter and terminates STSTATUS.EXE, a program that is necessary to run the progress meter.

```
iRVal =EndProgress( )
```

StopProgress Function

Description: Closes the progress meter without exiting STSTATUS.EXE.

Syntax: StopProgress()

StopProgress Function

Returns: Integer. FALSE (0) on error, otherwise TRUE (-1).

Remarks: No parameters. Use this function before the conclusion of the function whose progress is being measured. Use StopProgress to close the status meter when you will be restarting it immediately.

Example: This example closes the progress meter without terminating STSTATUS.EXE, a program that is necessary to run the progress meter.

```
iRetVal =StopProgress( )
```

Open and Save Dialog Box Functions You can display customized Open, Save, and Save As dialog boxes. These functions are declared for you in the WINCMD.SBH file. You can include this file and access these functions in any SBL script.

OpenDlg Function

Description: Creates a simple Open File dialog box.

Syntax: OpenDlg(sFile)

Returns: Integer. FALSE (0) on error, otherwise TRUE (-1).

Remarks: *sFile* is the filename returned from the Open dialog. This can also be the default value. This function lists all files. The dialog box caption is "Open".

Example: iRetVal = OpenDlg(sRetValFilename)

OpenDialog Function

Description: Creates a customized Open File dialog box.

Syntax: OpenDialog(sFile, sFilter, sCaption, hWnd, Flags)

OpenDialog Function

Returns: Integer. FALSE (0) on error, otherwise TRUE (-1).

Remarks: *sFile* is the filename returned from the Open dialog. This can also be the default value.

sFilter gives the text for the name of each filtering option to display in the dialog box, as well as the filtering criteria. If multiple criteria are used with any filtering option, separate the criteria with semicolons. Complete filtering options (text and criteria) are separated by commas.

sCaption is the dialog box title.

hWnd is the handle of the script or application calling the function.

flags are customizing options you can send to the dialog box. The most common are listed here; refer to any Windows API reference for a complete listing of available flags.

OFN_HIDEREADONLY – This *flag* hides the Read Only checkbox and the Help button on the dialog box.

OFN_PATHMUSTEXIST – This *flag* causes the dialog box to generate a message box if the selected path does not exist.

OFN_FILESMUSTEXIST – This *flag* restricts the user to typing names of existing files in the File Name field of the dialog box. Typing a nonexistent filename causes a warning to display in a message box. Using this *flag* automatically sets the OFN_PATHMUSTEXIST *flag*.

OFN_NOCHANGEDIR – This *flag* causes the system's current directory to revert to its setting before the function was called.

Example: The following example opens the Open dialog box with the caption "Select a Program to Run", and has three options in the file type drop-down list: Batch Files and Programs, Batch Files, and Programs.

```
hWnd = GetFocus( )
sFileTypes = "Batch Files and Programs (*.bat *.exe),*.bat;*.exe,Batch Files (*.bat),*.bat,Programs (*.exe),*.exe,"
iRVal = OpenDialog( sRetFilename, sFileTypes, "Select a Program to Run", hWnd, OFN_HIDEREADONLY)
```

SaveAsDlg Function

Description: Creates a simple Save As dialog box.

Syntax: SaveAsDlg(sFile)

Returns: Integer. FALSE (0) on error, otherwise TRUE (-1).

Remarks: This function lists all files and displays the caption "Save As".

Example: This example opens the Save As dialog box and stores the selected file in the variable sRetFilename.

```
iRetVal = SaveAsDlg( sRetFilename)
```

SaveAsDialog Function

Description: Creates a customized Save As dialog box.

Syntax: SaveAsDialog(sFile, sFilter, sCaption, hWnd, Flags)

Returns: Integer. FALSE (0) on error, otherwise TRUE (-1).

SaveAsDialog Function

Remarks: *sFile* is the filename returned from the Save As dialog. This can also be the default value.

sFilter gives the text for the name of each filtering option to display in the dialog box, as well as the filtering criteria. If multiple criteria are used with any filtering option, separate the criteria with semicolons. Complete filtering options (text and criteria) are separated by commas.

sCaption is the dialog box title.

hWnd is the handle of the script or application calling the function.

flags are customizing options you can send to the dialog box. The most common are listed here; refer to any Windows API reference for a complete listing of available flags.

OFN_HIDEREADONLY – This *flag* hides the Read Only checkbox and the Help button in the dialog box.

OFN_PATHMUSTEXIST – This *flag* causes the dialog box to generate a message if the selected path does not exist.

OFN_OVERWRITEPROMPT – This *flag* causes the display of a warning message and confirmation prompt if the chosen filename already exists in the selected path.

OFN_NOCHANGEDIR – This *flag* causes the system's current directory to revert to its setting before the function was called.

Example: The following example opens the Save As dialog box with the caption "Save File As", and limits the displayed file type to text files. It uses the value in *hWnd* as the handle to the parent window, restricts the user to save the file in a path that already exists, warns if the filename already exists, and returns the saved filename to the variable *sRetFilename*.

```
hWnd = GetFocus( )
sFileTypes = "Text Files (*.txt),*.txt,"
iRVal = SaveAsDialog( sRetFilename, sFileTypes, "Save File As",
hWnd, OFN_PATHMUSTEXIST + OFN_OVERWRITEPROMPT)
```

Calendar Functions Three calendar-related functions are available. These functions are declared for you in the WINCMD.SBH file. You can include this file and access these functions in any SBL script.

DaysPerWeek Function

Description: Returns the number of workdays per week for the Global Calendar.

Syntax: DaysPerWeek(PrmProject)

Returns: Integer giving the number of workdays per week.

Remarks: *PrmProject* is the currently bound PrmProject object. The PrmProject object must be bound using the Bind, Open, or Create function before calling DaysPerWeek.

Example: This example calls the DaysPerWeek function and displays the response in a message box.

```
iNumDays = DaysPerWeek( PrmProj)
sNumDays = cstr( iNumDays)
MsgBox "There are " & sNumDays & " per week", 0, "SureTrak Project Manager"
```

HoursPerDay Function

Description: Returns the number of hours per day as set in the Global Calendar.

Syntax: HoursPerDay(PrmProject)

Returns: Integer giving the number of workhours per day.

Remarks: *PrmProject* is the currently bound PrmProject object. The PrmProject object must be bound using the Bind, Open, or Create function before calling DaysPerWeek.

Example: This example calls the HoursPerDay function and displays the response in a message box.

```
iNumHours = HoursPerDay( PrmProj)
sNumHours = cstr( iNumHours)
MsgBox "There are " & sNumHours & " per day", 0, "SureTrak Project Manager"
```

StartOfWeek Function

Description: Returns the first workday of the week as set in the Global Calendar or in STWIN.INI.

Syntax: StartOfWeek(PrmProject)

Returns: Integer representing the first workday of the week.

Remarks: *PrmProject* is the currently bound PrmProject object. The PrmProject object must be bound using the Bind, Open, or Create function before calling DaysPerWeek.

The workday equivalents are

- | | |
|---------------|--------------|
| 1 – Sunday | 5 – Thursday |
| 2 – Monday | 6 – Friday |
| 3 – Tuesday | 7 – Saturday |
| 4 – Wednesday | |

Example: This example finds the first workday and, if it is a Monday, displays a message box.

```
iDay = StartOfWeek( PrmProj)
If iDay = 2 Then MsgBox "Monday", 0, "SureTrak Project Manager"
EndIf
```

Windows API Functions You can use any Windows API function in an SBL script. For your convenience, a few of the most common have been declared for you in the WINCMD.SBH file. You can include this file and access these functions in any SBL script.

GetFocus Function

Description: Gets the handle of the window that has input focus.

Syntax: GetFocus()

Returns: Integer giving the active window handle.

Remarks: None.

Example: This example returns the handle to the window with the focus.

```
HWnd = GetFocus( )
```

SetFocus Function

Description: Sets the focus to the window with the given handle.

Syntax: SetFocus(hWnd)

Returns: Integer. FALSE (0) on error, otherwise TRUE (-1).

Remarks: *hWnd* is the handle of the window to which the focus is being set.

Example: iRVal = SetFocus(hWnd)

FindWindow Function

Description: Retrieves the handle of the window whose ClassName and WindowName match the specified strings.

Syntax: FindWindow(lpClassName, lpWindowName)

FindWindow Function

Returns: Integer with the handle of the window.

Remarks: *lpClassName* is the class name of the window you are looking for.
lpWindowName is the exact name of the window you are looking for.

Example: This example returns the handle for Windows Explorer.

```
iHwnd = FindWindow( "ExploreWClass", "Exploring - My Computer")
```

SendMessage Function

Description: Sends a message to the specified window.

Syntax: SendMessage(hWnd, wParam, lParam, lParam)

Returns: Integer. FALSE (0) on error, otherwise TRUE (-1).

Remarks: *hWnd* is the handle of the window to receive the message.

wMsg is the message to send.

wParam specifies 16 bits of additional message-dependent information.

lParam specifies 32 bits of additional message-dependent information.

Example: This example sends the WM_PM_SETTEXT message to the window whose handle is in hWndMeter:

```
SendMessage( hWndMeter, WM_PM_SETTEXT, TextHwnd, 0)
```

EnableWindow Function

Description: Enables or disables mouse or keyboard input to a given window.

Syntax: EnableWindow(hWnd, bEnable)

Returns: Integer. FALSE (0) on error, otherwise TRUE (-1).

Remarks: *hWnd* is the handle of the window to enable or disable.
bEnable specify True to enable; False to disable.



Remember to re-enable any window you have disabled before exiting the script, or the window will remain disabled.

Example: This example disables SureTrak:

```
hWnd = STWApp.HwndiRVa1 = EnableWindow( hWnd, False)
```

GetActiveWindow Function

Description: Gets the handle of the active window.

Syntax: GetActiveWindow()

Returns: Handle of the active window.

Remarks: This function is similar to GetFocus, however GetActiveWindow returns the handle even if the active window does not have the input focus.

Example: `hWnd = GetActiveWindow()`

SetWindowPos Function

Description: Changes the size, position, and Z-order of child, pop-up, and top-level windows. Windows are ordered by their appearance onscreen: the window on top receives the highest rank and is the first window in the Z-order.

Syntax: SetWindowPos(hWnd, hWndInsertAfter, x, y, cx, cy, wFlags)

Returns: Integer. FALSE (0) on error, otherwise TRUE (-1).

Remarks: *hWnd* is the handle of the window to be positioned.

hWndInsertAfter is the handle of the window to precede the positioned window in the Z-order: this can also be HWND_BOTTOM, HWND_TOP, or HWND_TOPMOST.

x specifies the new position of the left edge of the window.

y specifies the new position of the top edge of the window.

cx specifies the new width of the window.

cy specifies the new height of the window.

wFlags control how the window is positioned.

Example: This example moves SureTrak to the top. Since the flags specify not to move or resize the window, it remains in the same position.

```
hWnd = STWApp.Hwnd  
CALL SetWindowPos( hWnd, HWND_TOP, 0, 0, 0, 0, SWP_NOMOVE OR SWP_NOSIZE)
```

GetPrivateProfileString Function

Description: Reads information from a Windows initialization file such as STWIN.INI.

Syntax: GetPrivateProfileString(lpSectionName, lpKeyName, lpDefault, lpReturnedString, nSize, lpFileName)

Returns: Integer. FALSE (0) on error, otherwise TRUE (-1).

Remarks: *lpSectionName* is the name of the section in the INI file.

lpKeyName is the name of the key value in the INI file.

lpDefault is the default value to return if the key doesn't exist.

lpReturnedString is the value returned from the INI file.

Size is the number of characters to return.

lpFileName is the name of the INI file.

Example: Assume the INI file contains the following lines:

```
[ABC123]TestKey = Something
```

The following example reads the value "Something" into the variable sRetVal, unless the key TestKey does not exist, in which case it returns the value "Nothing":

```
sSectionName = "Abc123"
```

```
sKeyName = "TestKey"
```

```
sDefault = "Nothing"
```

```
INIFileName = "STWIN.INI"
```

```
iRVal = GetPrivateProfileString( sSectionName, sKeyName, sDefault, sRetVal,  
256, INIFileName)
```


GetPrivateProfileInt Function

Description: Reads information from a Windows initialization file such as STWIN.INI.

Syntax: GetPrivateProfileInt(lpSectionName, lpKeyName, nDefault, lpFileName)

Returns: Integer value.

Remarks: *lpSectionName* is the name of the section in the INI file.
lpKeyName is the name of the key value in the INI file.
nDefault is the default value to return if the key doesn't exist.
lpFileName is the name of the INI file.

Example: Assume the INI file contains the following lines:

```
[ABC123]
```

```
TestKey = 100
```

The following example reads the value 100 into the variable iRetVal:

```
sSectionName = "Abc123"
```

```
sKeyName = "TestKey"
```

```
iDefault = 0
```

```
INIFilename = "STWIN.INI"
```

```
iRetVal = GetPrivateProfileInt( sSectionName, sKeyName, iDefault, INIFilename)
```

WritePrivateProfileString Function

Description: Writes information to a Windows initialization file such as STWIN.INI.

Syntax: GetPrivateProfileString(lpApplicationName, lpKeyName, lpString, lpFileName)

Returns: Integer. FALSE (0) on error, otherwise TRUE (-1).

Remarks: *lpApplicationName* is the name of the section in the INI file.

lpKeyName is the name of the key value in the INI file.

lpString is the information to write to the INI file.

lpFileName is the name of the INI file.

Example: The following example writes "Something" to the TestKey in the [Abc123] section of STWIN.INI:

```
sSectionName = "Abc123"  
sKeyName = "TestKey"  
INIFilename = "STWIN.INI"  
iRetVal = WritePrivateProfileString( sSectionName, sKeyName, "Something",  
INIFilename)
```

DeletePrivateProfileString Function

Description: Deletes the specified key and value from a Windows initialization file such as STWIN.INI.

Syntax: DeletePrivateProfileString(lpApplicationName, lpKeyName, lpString, lpFileName)

Returns: Integer. FALSE (0) on error, otherwise TRUE (-1).

Remarks: *lpApplicationName* is the name of the section in the INI file.

lpKeyName is the name of the key value in the INI file.

lpString must be set to 0 to perform deletion.

lpFileName is the name of the INI file.

Example: The following example deletes the key "TestKey" and its associated value from the [Abc123] section of STWIN.INI:

```
sSectionName = "Abc123"  
sKeyName = "TestKey"  
INIFilename = "STWIN.INI"  
iRetVal = DeletePrivateProfileString( sSectionName, sKeyName, 0, INIFilename)
```

Event Triggers SureTrak exposes several event triggers which are activated by setting up "hooks" using a script written in SBL. A hook is a compiled or uncompiled function or subroutine in an SBL script that, once registered with SureTrak, is called automatically whenever a given event occurs.

You can register either a subroutine or a function as a hook for all events. However, if you use a subroutine as a hook where the event requires a return value, SureTrak will continue as if the value returned was TRUE. The following events are available in SureTrak:

- **ON_BEFORE_CLOSE.** Called when SureTrak is about to terminate. Return TRUE if SureTrak should terminate, FALSE if SureTrak should not terminate. Returning FALSE also aborts further processing of ON_BEFORE_CLOSE hooks.
- **ON_AFTER_CLOSE.** Called after SureTrak has terminated but before it unloads SBL. This event should be used for terminating or cleaning up after your own application. No return value is required; however, returning FALSE aborts further processing of ON_AFTER_CLOSE hooks.
- **ON_BEFORE_SCHEDULE.** Called before a project is scheduled. Return TRUE if the project should be scheduled, FALSE if the project should not be scheduled. Returning FALSE also aborts further processing of ON_BEFORE_SCHEDULE hooks.
- **ON_AFTER_SCHEDULE.** Called after a project is scheduled. No return value is required; however, returning FALSE aborts further processing of ON_AFTER_SCHEDULE hooks.
- **ON_BEFORE_LEVEL.** Called before a project is leveled. Return TRUE if the project should be leveled, FALSE if the project should not be leveled. Returning FALSE also aborts further processing of ON_BEFORE_LEVEL hooks.
- **ON_AFTER_LEVEL.** Called after a project is leveled. No return value is required; however, returning FALSE aborts further processing of ON_AFTER_LEVEL hooks.

- **ON_BEFORE_UPDATE.** Called before a project is updated. Return TRUE if the project should be updated, FALSE if the project should not be updated. Returning FALSE also aborts further processing of ON_BEFORE_UPDATE hooks.
- **ON_AFTER_UPDATE.** Called after a project is updated. No return value is required; however, returning FALSE aborts further processing of ON_AFTER_UPDATE hooks.
- **ON_BEFORE_POPEN.** Called when an existing project is about to be opened. Return TRUE if the project should be opened, FALSE if the project should not be opened. Returning FALSE also aborts further processing of ON_BEFORE_POPEN hooks.
- **ON_AFTER_POPEN.** Called after a project is opened. No return value is required; however, returning FALSE aborts further processing of ON_AFTER_OPEN hooks.
- **ON_BEFORE_PNEW.** Called when a new project is about to be created. Return TRUE if the project should be created, FALSE if the project should not be created. Returning FALSE also aborts further processing of ON_BEFORE_PNEW hooks.
- **ON_AFTER_PNEW.** Called after a project is created. No return value is required; however, returning FALSE aborts further processing of ON_AFTER_PNEW hooks.
- **ON_BEFORE_PSAVE.** Called when a project is about to be saved. Return TRUE if the project should be saved, FALSE if the project should not be saved. Returning FALSE also aborts further processing of ON_BEFORE_PSAVE hooks.
- **ON_AFTER_PSAVE.** Called when a project is saved. No return value is required; however, returning FALSE aborts further processing of ON_AFTER_PSAVE hooks.
- **ON_BEFORE_PCLOSE.** Called when a project is about to close. Return TRUE if SureTrak should close the project, FALSE if SureTrak should not close the project. Returning FALSE also aborts further processing of ON_BEFORE_PCLOSE hooks.
- **ON_AFTER_PCLOSE.** Called after a project is closed. No return value is required; however, returning FALSE aborts further processing of ON_AFTER_PCLOSE hooks.

- **ON_PROJECT_ACTIVATE.** Called when a project becomes the active project (top-most window). No return value is required; however, returning FALSE aborts further processing of ON_PROJECT_ACTIVATE hooks.
- **ON_AFTER_NEWPROJWIZ.** Called after a new project is created using Project KickStart. No return value is required; however, returning FALSE aborts further processing of ON_AFTER_NEWPROJWIZ hooks.
- **ON_SWITCH_TO_PERTVIEW.** Called after the view is changed to PERT from the Bar chart. No return value is required; however, returning FALSE aborts further processing of ON_SWITCH_TO_PERTVIEW hooks.
- **ON_SWITCH_TO_GANTVIEW.** Called after the view is changed to the Bar chart from PERT. No return value is required; however, returning FALSE aborts further processing of ON_SWITCH_TO_GANTVIEW hooks.
- **ON_AFTER_PASTE.** Called after pasting activities from the Clipboard. No return value is required; however, returning FALSE aborts further processing of ON_AFTER_PASTE hooks.

Each time you register a hook for an event, the hook is assigned a sequence number. When the event is triggered, the hooks are processed according to sequence number, beginning with 1. With all events except ON_BEFORE_CLOSE, the full pathname of the project is passed in as the first and only argument to the subroutine or function. You can choose to ignore the parameter; however, if you accept the parameter, it must be declared as a String. You can also accept a String as a parameter to the hook for an ON_AFTER_CLOSE event; however, the argument will always be an empty string. Functions must be returned as Long.

Listed below are the Basic functions used to register and delete hooks. These functions are available in any PRM object. All hooks are registered globally, and may be modified by any valid PRM object at any time.

EndEvent Function

Syntax: PrmObject.EndEvent(event, script, [subroutine])

Returns: Integer. FALSE (0) on error, otherwise TRUE (-1).

Remarks: The EndEvent function removes the registration of a script. Valid events are listed in PRMCONST.SBH.

Example: The following example uses the EndEvent function to unregister the foo script for the ON_PROJECT_CLOSE *event*:

```
prmobj.EndEvent( ON_PROJECT_CLOSE, "C:\STWIN\SCRIPTS\F00.SBX")
```

GetEvent Function

Syntax: PrmObject.GetEvent(event, [script], [subroutine])

Returns: Integer value indicating the sequence number for the *script* within the *event*. A sequence number of 0 indicates that no scripts are registered or that an error occurred.

Remarks: The GetEvent function retrieves the sequence number for a previously registered hook. Use the GetEvent function with the same arguments used when the event hook was created. If no script name is given, SureTrak returns the number of scripts currently registered for the specified *event*. Valid *events* are listed in PRMCONST.SBH.

Example: The following example uses the GetEvent function to test whether the foo script is already registered for the ON_CLOSE event by requesting its sequence number. If the foo script is not registered, it will be registered using the OnEvent function:

```
sequence% = prmobj.GetEvent( ON_CLOSE, "C:\STWIN\SCRIPTS\F00.SBX")
If( sequence = 0) Then
  prmobj.OnEvent( ON_CLOSE, "C:\STWIN\SCRIPTS\F00.SBX")
End If
```

OnEvent Function

Syntax: PrmObject.OnEvent(event, script, [subroutine], [sequencenumber])

Returns: The sequence for the script within the *event*. A sequence number of 0 indicates that an error occurred.

Remarks: The OnEvent function registers a script to be executed when a specified *event* occurs. Arguments to OnEvent are

event Integer value identifying the SureTrak *event* to register. Valid values are defined in PRMCONST.SBH.

script String value indicating the name of the file containing the subroutine to run when the *event* occurs.

subroutine String value indicating the name of the subroutine or function to run within the script file. If this value is omitted, the main subroutine is run.

sequencenumber Integer identifying the requested sequence position to use for this event. If the sequence position is used by another event script, this event is inserted at the requested sequence position, and all subsequent event scripts move to the next higher sequence position. If a sequence position lower than *sequencenumber* is available, this event is inserted at the lower sequence position. The first sequence position is 1.

You can register an event hook by creating any SureTrak Prm object and then calling the object's OnEvent function. Any time the event occurs in SureTrak after this function has been called, the hook is run. By default, all hooks are given the next available sequence number.

Example: The following example uses the OnEvent function to register the foo script if it is not already registered for the ON_CLOSE event; the GetEvent function is used to determine whether or not the script has already been registered:

```
sequence% = app.GetEvent( ON_CLOSE, "C:\STWIN\SCRIPTS\F00.SBX")
If( sequence = 0) Then
app.OnEvent( ON_CLOSE, "C:\STWIN\SCRIPTS\F00.SBX")
End If
```

RunEvent Function

Syntax: PrmApp.RunEvent(event, [arguments])

Returns: The return value for the RunEvent function is the return value from the script associated with the *event* (TRUE for scripts which do not return a value). If more than one script is registered for the *event*, the return value is the value returned by the last script.

Remarks: The RunEvent function executes all the scripts registered with an *event* without actually triggering it. *Arguments* should be the name of a project unless the *event* is an ON_BEFORE_CLOSE event, in which case *arguments* should be an empty string. Valid *events* are listed in PRMCONST.SBH.

Example: The following example uses the RunEvent function to run all scripts associated with the ON_CLOSE event

```
app.RunEvent( ON_CLOSE)
```

To set a hook when SureTrak first opens, create a script with a main subroutine. The main subroutine should call OnEvent for each hook you want to register. Then, append the initializing script name (including the .SBL or .SBX extension) as an argument to the SureTrak command line, or include it in the [Startup] section of STWIN.INI. SureTrak accepts multiple arguments, processing them left to right.

In all cases, if a hook returns an abort value (FALSE), subsequent hooks for that event are not processed. For example, if an ON_BEFORE_PCLOSE hook returns FALSE to abort the close, subsequent ON_BEFORE_PCLOSE hooks are not processed.

Open/SaveAs Extensions The project type list in the Open Project and Save As dialog boxes can be extended to read and write other formats. Add a new project type to the project type list by adding an entry to the STWIN.INI file, located in your \WINDOWS directory. The actual conversion of the data is done through an SBL script or DLL. Each of these methods is described in the following sections.

To add project types to the Open/Save As drop-down lists

To enable an Open/Save As extension, add an entry to the STWIN.INI file in your \WINDOWS directory. This entry provides information SureTrak needs to call your script or DLL when opening a project of the new type.

To extend the project type list, add a new section called [Imports] to your STWIN.INI file. In this section, add the new type using the following format:

```
type name = script or dll, type extension
```

Type name is the text that is listed in the Type field. *Script or dll* is the name of the SBL, SBX, or DLL file which will read or write the project data; the full path and name of the script or DLL, including extension, are required. *Type extension* is the file extension SureTrak uses to identify valid projects.

For example, add the following line for an SBL script that opens a Text File project; the script name is OPENTEXT.SBL, and the project extension is TXT:

```
Text File = C:\STWIN\SCRIPTS\OPENTEXT.SBL,TXT
```

Add the following line for a DLL that opens a Text File project; the script name is OPENTEXT.DLL and the project extension is TXT:

```
Text File = C:\STWIN\OPENTXT.DLL,TXT
```

To extend the SureTrak project types using SBL

To extend the project type list in the Open and Save As dialog boxes using an SBL script, create a script file that contains one or more of the following functions. Include functions based on the capabilities that you want to support. To support only reading of the new project type, you need only include a PrmRead function. To support only writing of the new project type, include a PrmWrite function. To support both reading and writing of the new project type, you need to include PrmRead and PrmWrite functions.



Declare all functions as Long.

- **Function PrmValid(*name* As String) As Long**

When a user selects the type in the Type field, SureTrak creates a list of all files with the given extension in the current path. For each of these files, SureTrak calls the PrmValid function. If the returned value is TRUE, SureTrak displays the project and project information in the project listing. If the returned value is FALSE, SureTrak ignores the file.

Use this function to verify that the project is a valid project of the type you are supporting. *Name* is the name of the project that will be opened or saved.

If no PrmValid function is defined, SureTrak assumes that all the files with the specified extension are valid projects.

- **Function PrmRead(*p* As PrmProject) As Long**

After the user selects a project in the Open Project dialog box, SureTrak creates a default project using the TEMPLATE project files. If TEMPLATE does not exist, internal defaults are used. SureTrak then calls the PrmRead function to load the data for the project being converted.

Use this function to read the project information for the type you are supporting. *p* is a valid PrmProject object. Use the PrmProject, AddNew, and Put methods to read your project information into the open project.

If the return value is TRUE, SureTrak opens the project; otherwise, SureTrak aborts the opening.

If no PrmRead function is defined, SureTrak displays a message indicating that no Open capabilities were found.

- **Function PrmWrite(*p* As PrmProject) As Long**

After the user selects a project in the Save As dialog box, SureTrak calls the PrmWrite function to save the project information.

Use this function to save the project information for the type you are supporting. *p* is a valid PrmProject object. Use the PrmProject Get method to retrieve your project information from the open project.

If the return value is FALSE, SureTrak aborts the save.

If no PrmWrite function is defined, SureTrak displays a message indicating that no Save As capabilities were found.

To extend SureTrak project types using a DLL

To extend the project type list in the Open and Save As dialog boxes using a DLL, you must create a Windows DLL that exports one or more of the following functions. Include functions based on the capabilities you want to support. To support only reading of the new project type, you need only include a `prmread` function. To support only writing of the new project type, you need only include a `prmwrite` function. To support reading and writing of the new project type, include the `prmread` and a `prmwrite` functions.



Functions must be exported by name. Exporting using ordinal values is not supported.

- **`int prmvalid(LPCSTR filename, int filenumber)`**

When a user selects the project type in the Type field in the Open or Save As dialog box, SureTrak creates a list of all files that have the given extension in the current path. For each of these files, SureTrak calls the `prmvalid` function in the conversion DLL. `Prmvalid` may also be called for each type of database file (identified by the `filenumber` parameter).

`Filename` is the name of the file found by SureTrak. `Filenumber` is the ID of the database file requested by SureTrak. File and field IDs are listed in `PRMCONST.SBH`.

Use `prmvalid` to verify that all project database files are valid, returning `FALSE` if any file is invalid. If the return value is `TRUE`, SureTrak displays the project group name and project information in the project list. If the return value is `FALSE`, SureTrak ignores the file(s).

If no `prmvalid` function is exported, SureTrak assumes that all the files with the given extension are valid projects.

- **int prmread**(LPCSTR filename, int filenumber)

After the user selects a project using the Open Project dialog box, SureTrak calls the prmread function to load the data for the project being converted.

Filename is the name of the project file(s) being read. *filenum* is the ID of the database file requested by SureTrak. File and field IDs are listed in PRMCONST.SBH.

Use this function to read the project information for the project type you are supporting. SureTrak calls *prmread* for each type of database file (identified by the *filenumber* parameter). The function returns the number of records read or an error code. Error codes are defined in STWFILES.HPP. If you want SureTrak to open the default file, from the TEMPLATE project or the default file for your project, return an error code of RUN_DEF (-1).

If no prmRead function is exported, SureTrak displays a message indicating that no Open capabilities were found.

- **int prmwrite**(LPCSTR filename, int filenumber)

After the user selects a project in the Save As dialog box, SureTrak calls the prmwrite function to save the project information.

Filename is the name of the file found by SureTrak. *Filenumber* is the ID of the database file requested by SureTrak. Database file IDs are defined in STWFILES.HPP.

Use this function to save the project information for the project type you are supporting. SureTrak calls prmwrite for each type of database file (identified by the filenumber parameter). The function returns 1 if successful. Return a value of 0 or an error code to abort the save. Error codes are defined in STWFILES.HPP. If you want SureTrak to save the default file as part of your project, return an error code of RUN_DEF (-1).

If no prmWrite function is exported, SureTrak displays a message indicating that no Save As capabilities were found.

- **int prminit**(int filename, int flag)

SureTrak calls the prminit function at specific intervals during the open and save processes so you can perform any additional processing or initialization.

Filename is the name of the file found by SureTrak. *Flag* is a value identifying the current state of the process (for example, before reading or after writing).

Use this function to perform any additional processing or initialization that needs to be performed while opening or saving your project. SureTrak calls prminit with the *flag* values corresponding to the states listed in the following table.

Flag value	State
FIO_BEFORE_READ	Immediately before a call to prmread.
FIO_MIDDLE_READ	Immediately after each call to prmread.
FIO_BEFORE_FA2	After all files have been read, but before the activities have been organized for display in the project window.
FIO_AFTER_READ	After all files have been read and all project indexes have been built.
FIO_BEFORE_WRITE	Immediately before a call to prmwite.
FIO_AFTER_WRITE	After all files have been saved.
FIO_FAILED_WRITE	If saving of files fails.

By exporting this function you can perform any necessary initialization and clean up. If you return FALSE, the current process is aborted. If no prminit function is exported, SureTrak assumes a TRUE return value for all states.

Valid *flags* are defined in STWFILES.HPP.

Default Files Default files are project files in the SureTrak file format. One default file exists for each database file ID. These files are created by opening a copy of the corresponding TEMPLATE project file stored in your \STWIN directory. You can have SureTrak create a copy of the TEMPLATE project file by returning RUN_DEF from a call to *prmread* for the corresponding database file ID.

When the *prmwrit*e function is called, if you return a value of RUN_DEF, SureTrak saves the copy of the TEMPLATE file, including any modifications made while the project was open, using the eight-character name of your project and the SureTrak file extension for the corresponding database file ID. You can then open this default file by returning RUN_DEF in *prmread* during subsequent openings of your project. This approach enables you to keep any SureTrak and project specific settings and modifications without saving them as part of the file format for your project. This feature can be especially useful when dealing with layout database files (FLAY).

By returning RUN_DEF on a call to *prmread* with a *filenumber* of FLAY, SureTrak will open a copy of the TEMPLATE projects layout file, unless a layout file in the same directory as your project has the same eight-character filename and a .LAY file extension; in this case, SureTrak opens that layout file. As you change the layout in your project, SureTrak records the changes as part of the layout. Then, by returning RUN_DEF during a subsequent call to *prmwrit*e with a *filenumber* of FLAY, SureTrak will save a copy of the layout using the eight-character name of your project and a .LAY extension. This layout file contains any changes and modifications made to the layout while your project was open.

SureTrak stores and retrieves individual records using two additional functions. When the new type is selected in the Type field in the Open or Save As dialog boxes, SureTrak calls the *prmprocs* registration function in your DLL. Define this function, which must be exported, as follows:

```
#include <windows.h>
typedef void( WINAPI* GETPROC )( LPVOID, int, int, LPCSTR, int);
typedef void( WINAPI* PUTPROC )( LPVOID, int, int, LPCSTR, int);
GETPROC GetRec;
PUTPROC PutRec;
LPVOID Obj;
void prmprocs( GETREC get, PUTREC put, LPVOID o)
{
    GetRec = get;
    PutRec = put;
    Obj = o;
}
```

The parameters to the *prmprocs* function contain pointers to unexported SureTrak functions that put and get individual records. In the *prmread* function, you will need to add each of the records by calling

```
PutRec( Obj, filenum, recordnum, recordstruct, 100);
```

filenum is the file you are adding records to. *recordnum* is a linear count of the record (0 through n, where n = number of records - 1). Currently, records must be added in sequential order (that is, you must add 0 before you add 1), but they needn't be in any given sort order. *recordstruct* is a pointer to the c-record structure containing the record information. Record structures are defined in STWFILES.HPP. The last parameter is a structure type number. Currently, the structures are all numbered 100. If the file structures change in the future, SureTrak automatically converts the old structure formats to the new ones to provide compatibility for changes in file structures.

In the *prmwrite* function, you will need to extract each record by calling

```
GetRec( Obj, filenum, recordnum, recordstruct, 100);
```

The arguments for this are the same as for *PutRec*. The *recordstruct* is filled with the data for the given record by SureTrak. SureTrak returns a structure of NULL values when it reaches the end of the file.