

Rainlendar_ChangeScheduleDate

Syntax

```
text = Rainlendar_ChangeScheduleDate(offset, window)
```

Parameters

- *offset* (integer) - The number of days to change the schedule's day. This can be negative.
- *window* (string) - The window which contains the schedule item.

Return

Nothing.

Description

Changes the day the schedule item shows.

Rainlendar_ConvertString

Syntax

```
text = Rainlendar_ConvertString(text, codepage)
```

Parameters

- *text* (string) - The text in some codepage.
- *codepage* (string) - The codepage which the text uses (e.g. "iso-8859-1", "windows-1258", "koi8-r", ...)

Return

The text converted to utf-8.

Description

Converts the given text to utf-8. All text that is given to Rainlendar must be in utf-8 format so if you get text e.g. from a web page which uses some different codepage you need to use this function to convert it first.

Rainlendar_CopyToClipboard

Syntax

```
text = Rainlendar_CopyToClipboard(text)
```

Parameters

- *text* (string) - The text to be copied.

Return

Nothing.

Description

Copies the given text to the clipboard.

Rainlendar_CreateComponent

Syntax

```
strUID = Rainlendar_CreateComponent(icalData, strCalendar)
```

Parameters

- *icalData* (string) - The event or task in iCalendar format (see RFC2445).
- *strCalendar* (string) - The name of the calendar where the event or task is added. If the parameter is not given or it is empty string the event or task is still created but it doesn't belong to any calendar and will disappear during the next refresh. (optional)

Return

The UID of the created event or task.

Description

Creates a new event or task. If the UID property is not given one will be created for the item automatically. If the UID is the same as one of the existing event or task the item will be overwritten.

Rainlendar_CreateTimer

Syntax

```
Rainlendar_CreateTimer(time, callback, userData)
```

Parameters

- *time* (integer) - The time span in milliseconds after which the callback gets executed.
- *callback* (function) - The callback function which is executed after the timeout.

- *userData* (string) - User data which is given to the callback. This parameter is optional.

Return
Nothing

Description

Creates a timer which will automatically run the callback function after the time has passed. The timer is ran continuously as long as the callback function returns True. Returning False will stop the timer. The callback is a normal lua function which takes the *userData* as its only parameter.

Rainlendar_DeleteComponent

Syntax

```
Rainlendar_DeleteComponent(strUid, strCalendar)
```

Parameters

- *strUid* (string) - The unique identifier of the event or task to be deleted.
- *strCalendar* (string) - The name of the calendar from where the event or task is deleted. If the parameter is not given or it is empty string the event or task is deleted from all calendars where it belongs. (optional)

Return
Nothing.

Description

Deletes the given event or task. The item can be deleted from some specific calendar or from all of them.

Rainlendar_DismissAlarm

Syntax

```
Rainlendar_DismissAlarm(uid)
```

Parameters

- *uid* (string) - The identifier of the event to be dismissed.

Return

Nothing

Description

Dismisses the alarm. The event or task needs to be in the alarm window or nothing happens.

Rainlendar_Download

Syntax

```
Rainlendar_Download(url, callback, userData)
```

Parameters

- *url* (string) - The url of the page to be downloaded.
- *callback* (function) - The callback function which is executed after the download has been done.
- *userData* (string) - User data which is given to the callback. This parameter is optional.

Return

Nothing

Description

Downloads a page from the web. This can be used to read information from the internet, parse it and display in the Rainlendar's windows. Note that the downloaded content is returned as a string so downloading binary data isn't possible (see [Rainlendar DownloadElement](#) for downloading images). The callback is a normal lua function which gets the result code, downloaded content and the userdata as parameters (*function NameOfTheCallback(result, content, userData)*). The result code is the HTTP response from the server (e.g. 200 if all went fine, 404 if the file was not found, ...)

Rainlendar_DownloadElement

Syntax

```
Rainlendar_DownloadElement(url, callback, element, userData)
```

Parameters

- *url* (string) - The url of the element (usually an image) to be downloaded.
- *callback* (function) - The callback function which is executed after the

download has been done.

- *element* (string) - The identifier of the element which is replaced with the downloaded file.
- *userData* (string) - User data which is given to the callback. This parameter is optional.

Return

Nothing

Description

Downloads a file (=image) from the net. After the file has been downloaded it is replaced as the element so the next time the window is redrawn (see [Rainlendar Redraw](#)) the old image is replaced with the new one. The callback similar lua function as in [Rainlendar Download](#) but the content parameter is always nil.

Rainlendar_EnableCalendar

Syntax

```
Rainlendar_EnableCalendar(calendar, enabled)
```

Parameters

- *calendar* (string) - The name of the calendar.
- *enabled* (string) - True, if the calendar should be enabled and false if it should be disabled.

Return

Nothing

Description

Enables or disables the given calendar. When a calendar is enabled it's contents are re-read and the new events and tasks are shown in the windows. Note that you need to call [Rainlendar Redraw](#) function before the changes will be visible.

Rainlendar_Execute

Syntax

```
Rainlendar_Execute(url)
```

Parameters

- *url* (string) - The url that is opened.

Return

Nothing

Description

Launches the given url in a default web browser.

Rainlendar_FlushComponents

Syntax

```
changes = Rainlendar_FlushComponents()
```

Parameters

None

Return

Nothing

Description

Flushes the pending changes in the component queue. Rainlendar flushes the queue automatically too so usually it's not necessary to call this except if you want the changes visible immediately. If this function returns true you should call [Rainlendar Redraw\(\)](#) to redraw the windows and show the changes.

Rainlendar_GetAlarmItems

Syntax

```
alarms = Rainlendar_GetAlarmItems()
```

Parameters

None

Return

A table of currently visible alarm item UIDs.

Description

Returns a table of event or task identifiers which are currently displayed in the alarm window.

Rainlendar_GetAllItems

Syntax

```
tableOfItems = Rainlendar_GetAllItems(window)
```

Parameters

- *window* (string) - The window which items are to be listed.

Return

A table of all items (buttons, images, lists, ...)

Description

Returns all the items which the window contains. Note that an 'item' is not event or a task but a graphical thing in the window (button, image, list, calendar, ...). The returned table contains the item identifiers as defined in the skin's xml file. The window name parameter must match to only one window for this function to work.

Rainlendar_GetComponent

Syntax

```
strIcalData = Rainlendar_GetComponent(strUid)
```

Parameters

- *strUid* (string) - The unique identifier of the event or task.

Return

The event or task in iCalendar format (see RFC2445).

Description

Returns the event or task in iCalendar format. You can then modify the data as you wish and call the [Rainlendar_UpdateComponent\(\)](#) to apply the changes.

Rainlendar_GetDisplayDate

Syntax

```
day, month, year = Rainlendar_GetDisplayDate()
```

Parameters

None

Return

Day, month and year of the currently visible month. The months start from 0 (=January),

Description

Returns the date of the currently visible month. Note that this is not the current date but the first day of the month the calendar is currently showing.

Rainlendar_GetElementValue

Syntax

```
value = Rainlendar_GetElementValue(skin, element, field)
```

Parameters

- *skin* (string) - The name of the skin.
- *element* (string) - The name of the element in the skin.
- *field* (string) - The name of the element's field.

Return

The field's current value.

Description

Returns the given element field's value. The field parameter is the name of the field as written in the skin's xml file. So, for example for bitmap elements you can use there fields: opacity, hue, saturation, value, brightness, contrast, frames. See the skinning reference documentation for more information what values the fields can have.

Rainlendar_GetItemValue

Syntax

```
value = Rainlendar_GetItemValue(window, item, field)
```

Parameters

- *window* (string) - The name of the window.
- *item* (string) - The name of the item in the window.
- *field* (string) - The name of the item's field.

Return

The field's current value.

Description

Returns the given item field's value. You can use the [Rainlendar_GetAllItems](#) function to get a list of all items in the window or just enter the identifier of the item if you already know it. The field parameter is the name of the field as written in the skin's xml file. So, for example for image items you can use there fields: x1, y1, x2, y2, w, h, origin1, origin2, visible, align, scaling, margins, element, tooltip, tooltipHeader, tooltipText, frame, maxsizefromimage, minsizefromimage. See the skinning reference documentation for more information what values the fields can have. The window name parameter must match to only one window for this function to work.

Rainlendar_GetOption

Syntax

```
value = Rainlendar_GetOption(path, name)
```

Parameters

- *path* (string) - The path to the setting. This is the text inside the brackets in the rainlendar2.ini file.
- *name* (string) - The name of the setting.

Return

The value from the settings.

Description

Returns the value directly from the Rainlendar's settings (i.e. from the rainlendar2.ini settings file). If the settings doesn't exist an empty string is returned. You can check what values you can use in the pat and name parameters by opening the rainlendar2.ini file with a text editor and checking what it contains.

Rainlendar_GetString

Syntax

```
text = Rainlendar_GetString(text)
```

Parameters

- *text* (string) - The text to be translated.

Return

Translated version of the text.

Description

Gets a translated version of the string. If the translation is not available the same string is returned.

Rainlendar_GetVariable

Syntax

```
value = Rainlendar_GetVariable(skin, window, name)
```

Parameters

- *skin* (string) - The name of the skin.
- *window* (string) - The name of the window.
- *name* (string) - The name of the variable.

Return

The current value of the variable.

Description

Returns the current value of the given variable. For global variables the window parameter can be left empty. If the variable cannot be found an empty string is returned.

Rainlendar_GetWindowPosition

Syntax

```
x, y = Rainlendar_GetWindowPosition(strWindow)
```

Parameters

- *strWindow* (string) - The name of the window.

Return

The x and y coordinate of the given window.

Description

Returns the position of the given window.

Rainlendar_GetWindowSize

Syntax

```
w, h = Rainlendar_GetWindowSize(strWindow)
```

Parameters

- *strWindow* (string) - The name of the window.

Return

The width and height of the given window.

Description

Returns the size of the given window.

Rainlendar_GetWindows

Syntax

```
windows = Rainlendar_GetWindows()
```

Parameters

None

Return

A table of all windows.

Description

Returns a table which contains the names of all the windows that are enabled. A window name consists of the name of the skin a double colon and the name of the window. E.g. "Shadow4::Event List". Note that also the hidden windows are returned. You can check the state of the window with [Rainlendar_IsWindowVisible](#) function.

Rainlendar_HideWindow

Syntax

```
Rainlendar_HideWindow(window)
```

Parameters

- *window* (string) - The window to be hidden.

Return

Nothing

Description

Hides the given window. If the window name matches multiple windows (e.g. if the skin prefix is not given) all the matched windows are hidden.

Rainlendar_IsCalendarEnabled

Syntax

```
Rainlendar_IsCalendarEnabled(calendar)
```

Parameters

- *calendar* (string) - The name of the calendar.

Return

True if the calendar is enabled and false if it is not.

Description

Returns the given calendar's state. Returns nothing if the calendar doesn't exist.

Rainlendar_IsWindowVisible

Syntax

```
visible = Rainlendar_IsWindowVisible(window)
```

Parameters

- *window* (string) - The window which visibility is to be checked.

Return

True if the window is visible and false if it is hidden.

Description

Returns the visibility status of the given window. If the window name matches multiple windows (e.g. if the skin prefix is not given) the first found window's visibility is checked.

Rainlendar_ListAllComponents

Syntax

```
array<HTD> = Rainlendar_ListAllComponents(calendar, data)
```

```
arrayUIDS = Rainlendar_ListAllComponents(strCalendar, strDate)
```

Parameters

- *strCalendar* (string) - The name of the calendar which events are listed. If the parameter is not given the events and tasks from all calendars are returned. (optional)
- *strDate* (string) - The date for which the events are returned. If the date parameter is given only the events (not tasks!) on that day are returned. You can also use day names like today, tomorrow, monday, ... (optional)

Return

A table of UIDs for all the events and tasks.

Description

Returns the events and task UIDs. You can use the [Rainlendar GetComponent\(\)](#) to get the actual data for the event/task. This function can be used to return all the events/task or just on a specific day.

Rainlendar_Log

Syntax

```
Rainlendar_Log(text)
```

Parameters

- *text* (string) - The text that is written to the log.

Return

Nothing

Description

Writes a log entry to Rainlendar's log file. This is useful function when debugging the Lua scripts.

Rainlendar_Md5

Syntax

```
md5 = Rainlendar_Md5(text)
```

Parameters

- *text* (string) - The text

• *text* (string) - The text.

Return

MD5 sum as a string.

Description

Calculates MD5 sum for the given text.

Rainlendar_Message

Syntax

```
result = Rainlendar_Message(text, flags)
```

Parameters

- *text* (string) - The text that is shown in the message box.
- *flags* (integer) - The flags which define what kind of message box dialog is show. The value can be one of the following:

Buttons:

Value	Description
4	Puts an Ok button on the message box.
10	Puts Yes and No buttons on the message box.
20	Puts an Ok and Cancel buttons on the message box.
26	Puts Yes, No and Cancel buttons on the message box.

Icons:

Value	Description
256	Displays an exclamation mark symbol.

512	Displays an error symbol.
1024	Displays a question mark symbol.
2048	Displays an information symbol.

Return

The return value is the button which the user clicked.

Value	Button
2	Yes
4	Ok
8	No
16	Cancel

Description

Opens a message box and waits until the user clicks one of the buttons. The message box can be used e.g. to show errors to the user or ask questions.

The icon and button values for the flags can be combined by adding the values together. For example if you want to have Yes and No buttons with the question mark icon you would use value 1034 (= 1024 + 10)

Rainlendar_OpenDialog

Syntax

```
Rainlendar_OpenDialog(dialog, param)
```

Parameters

- *dialog* (string) - The name of the dialog to be opened. The value can be one of

the following:

Value	Dialog
"About"	The about dialog.
"Event"	The event dialog for a new event.
"Todo"	The to do dialog for a new task.
"Manager"	The manager dialog.
"Options"	The options dialog.

- *param* (string) - The parameter for the dialog. For "Event" and "Todo" dialogs it can contain the UID of the item to be edited.

Return

Nothing

Description

Opens the given dialog. If the dialog is already open and there can be only one of them open at the same time (e.g Options dialog) the existing dialog is brought to front and activated.

Rainlendar_PlaySound

Syntax

```
Rainlendar_PlaySound(soundFile, loop)
```

Parameters

- *soundFile* (string) - The full path to the audio file.
- *loop* (integer) - Set to 1 to make the sound to loop until [Rainlendar_StopSound](#) is called. (optional)

Return

Nothing

Description

Plays the give audio file.

Rainlendar_QuitApplication

Syntax

```
Rainlendar_QuitApplication( )
```

Parameters

None

Return

Nothing

Description

Quits Rainlendar.

Rainlendar_Redraw

Syntax

```
Rainlendar_Redraw(update, window)
```

Parameters

- *update* (integer) - Set to 1 to redraw also the window background on copy transparency.
- *window* (string) - The name of the window to be redrawn. (optional)

Return

Nothing

Description

Redraws the window(s). If the *window* parameter is not given all windows are redrawn. The *update* parameter defines if the background is refreshed or not. You should always set it to 1 if you make modifications which might change the position or size of the window. This function (or [Rainlendar Refresh](#)) needs to be called after you make any kind of changes before they are shown in the windows.

Rainlendar_Refresh

Syntax

Rainlendar_Refresh()

Parameters

None

Return

Nothing

Description

Refreshes Rainlendar. The difference between refresh and redraw is that with refresh everything is recreated from scratch. This includes all windows, skins, scripts and calendars. Because also the scripts are reloaded it means that any data that a script might have stored to variables will be gone.

Rainlendar_ResetScheduleDate

Syntax

```
text = Rainlendar_ResetScheduleDate(window)
```

Parameters

- *window* (string) - The window which contains the schedule item.

Return

Nothing.

Description

Resets the day the schedule item shows to the current day.

Rainlendar_SetDisplayDate

Syntax

```
Rainlendar_SetDisplayDate(day, month, year)
```

Parameters

- *day* (integer) - Currently displayed day. At the moment this needs to be always 1.
- *month* (integer) - Currently displayed month. January = 0, February = 1, ...
- *year* (integer) - Currently displayed year.

Return

Nothing

Description

Changes the currently displayed date. You need to call the [Rainlendar Redraw](#) function to apply the change in the windows.

Rainlendar_SetElementValue

Syntax

```
Rainlendar_SetElementValue(skin, element, field, value)
```

Parameters

- *skin* (string) - The name of the skin.
- *element* (string) - The name of the element in the skin.
- *field* (string) - The name of the element's field.
- *value* (string) - The new value for the item's field.

Return

Nothing

Description

This sets a new value for the element's field. See [Rainlendar_GetElementValue](#) function for more details about the item and their fields. Note that you need to call [Rainlendar_Redraw](#) function before the changes will be visible.

Rainlendar_SetEventHandler

Syntax

```
Rainlendar_SetEventHandler(callback, event, window, userData)
```

Parameters

- *callback* (function) - The callback function which is executed after the download has been done.
- *event* (integer) - The code for the event (see below).
- *window* (string) - The name of the window. For global events this must be an empty string.
- *userData* (string) - User data which is given to the callback. This parameter is optional.

Return

Nothing

Description

Creates an event handler which executes the given callback when the event occurs. The callback is a normal lua function which gets an event data and the userdata as parameters (*function NameOfTheCallback(eventData, userData)*). The event data depends on the event which was handler. See the below tables for the values.

Possible values for the event parameter are:

Value	Event data	Description
1	MouseEvent	Left mouse down
2	MouseEvent	Left mouse up
3	MouseEvent	Middle mouse down
4	MouseEvent	Middle mouse up
5	MouseEvent	Right mouse down
6	MouseEvent	Right mouse up
7	MouseEvent	Mouse move
8	MouseEvent	Mouse enters the window
9	MouseEvent	Mouse leaves the window

10	MouseEvent	Left double click
11	MouseEvent	Middle double click
12	MouseEvent	Right double click
13	MouseEvent	Mouse wheel
1000	nil	Power resume event. Send when the computer resumes from suspend/hibernation. This is a global event.

1001

SyncEvent c

MouseEvent data is a table with this content:

Field	Type	Description
x	integer	The x-coordinate of the mouse.
y	integer	The y-coordinate of the mouse.
wheel	integer	The mouse wheel value.
button	integer	The mouse button state.

SyncEvent data is a table with this content:

Field	Type	Description
status	string	The current status ("ok", "sync", "offline") for the calendar.
calendar	string	The name of the calendar being accessed.

Rainlendar_SetItemValue

Syntax

```
Rainlendar_SetItemValue(window, item, field, value)
```

Parameters

- *window* (string) - The name of the window.
- *item* (string) - The name of the item in the window.
- *field* (string) - The name of the item's field.
- *value* (string) - The new value for the item's field.

Return

Nothing

Description

This sets a new value for the item's field. See [Rainlendar_GetItemValue](#) function for more details about the item and their fields. Note that you need to call [Rainlendar_Redraw](#) function before the changes will be visible.

Rainlendar_SetOption

Syntax

```
Rainlendar_SetOption(path, name, value)
```

Parameters

- *path* (string) - The path to the setting. This is the text inside the brackets in the rainlendar2.ini file.
- *name* (string) - The name of the setting.

- *value* (string) - The new value for the setting.

Return

Nothing

Description

Sets the value directly to the Rainlendar's settings (i.e. to the rainlendar2.ini settings file). Note that you can really mess up all settings with this function unless you know what you are doing so be careful when using it. The value is stored only to the settings file so you need to call [Rainlendar Refresh](#) (or wait until the user refreshed/restarts Rainlendar) before it is actually taken into use.

Rainlendar_SetVariable

Syntax

```
Rainlendar_SetVariable(skin, window, name, value)
```

Parameters

- *skin* (string) - The name of the skin.
- *window* (string) - The name of the window.
- *name* (string) - The name of the variable.
- *value* (string) - The new value for the variable.

Return

Nothing

Description

Sets a new value for the variable. The variable must be defined in the skin file before it's possible to set its value (i.e. you cannot use this function to create new variables).

Rainlendar_SetWindowPosition

Syntax

```
Rainlendar_SetWindowPosition(strWindow, x, y)
```

Parameters

- *strWindow* (string) - The name of the window to be repositioned.
- *x* (string) - The new x coordinate for the window.
- *y* (string) - The new y coordinate for the window.

Return

Nothing.

Description

Changes the position of the given window.

Rainlendar_ShowMenu

Syntax

```
index = Rainlendar_ShowMenu(x, y, menu)
```

Parameters

- *x* (integer) - The x position where the menu is opened.
- *y* (integer) - The y position where the menu is opened.
- *menu* (table) - An array of menu items (strings). If strings are used as keys the list will be in sorted by them.

Return

The selected index as a string. If the user doesn't select anything from the menu an empty string is returned.

Description

Opens a context menu on the given position. If the x and y are set to -1 the menu is opened on the current mouse position. The menu item with text "---" is converted to a separator. If the table value is a table too (" {SubMenuName={Item1, Item2, Item3}}") a sub menu is created from it.

Rainlendar_ShowWindow

Syntax

```
Rainlendar_ShowWindow(window)
```

Parameters

- *window* (string) - The window to be shown.

Return

Nothing

Description

Shows the given window. If the window name matches multiple windows (e.g. if the skin prefix is not given) all the matched windows are shown.

Rainlendar_SnoozeAlarm

Syntax

```
Rainlendar_SnoozeAlarm(uid, duration)
```

Parameters

- *uid* (string) - The identifier of the event to be dismissed.
- *duration* (int) - The number of seconds the alarm is snoozed.

Return

Nothing

Description

Snoozes the alarm for given amount of time. The event or task needs to be in the alarm window or nothing happens.

Rainlendar_StopSound

Syntax

```
Rainlendar_StopSound()
```

Parameters

None

Return

Nothing

Description

Stops the sound started with [Rainlendar PlaySound](#).

Rainlendar_UpdateComponent

Syntax

```
Rainlendar_UpdateComponent(icalData)
```

Parameters

• *icalData* (string) - The ical data to be updated. (see [RFC2445](#))

- *icalData* (string) - 1 ne event or task in iCalendar format (see RFC2445).

Return

Nothing

Description

Updates an existing event or task. If the UID property of the event or task has been changed a new item will be created. Otherwise the existing event or task is updated.

Rainlendar_Version

Syntax

```
Rainlendar_Version()
```

Parameters

None

Return

The version numbers.

Description

Returns the current version of Rainlendar. This returns 4 values. The first three are the version (e.g. 2, 4, 0) and the last is the build number (e.g. 69).