
CABIntegr General information

Cabinet (.cab) are used to deliver software packages onto a device. Normally, a CAB file contains all necessary files and an "instruction" on how to install them. This "instruction" may contain commands to create/change registry keys, to copy files. A CAB file may contain a special file (hereafter – setup.dll) which contains the install/uninstall scenario determined by a software developer. CABIntegr allows to create/modify CAB files, view their contents, extract data and perform installation.

Working with CABIntergr:

The basic working unit is a "project". A "project" is a folder containing all necessary files and a configuration file with the extension PSINF. The list of projects can be viewed via option "Project" of the Main menu. A CAB file is both the initial and the resulting object of working with the project. There are two options to create a new project:
1. To select the command "Create" in the Main menu.
2. To select the command 'CAB files' in the Main menu, select the needed file from the list of the existing files, and to launch the command "Edit" for it.
To generate a CAB file, the command "Create CAB" has to be selected in the project edit window. Projects are saved in a special workplace folder. It makes sense to move it to a storage card (done via settings). It is also advisable to move a folder for temporary files to a storage card. This is required because the main memory space of the device is limited.

Differences of CAB-files created with CABIntegr from files created with MS CabWiz:

1. The file .000 with service information used to provide backward compatibility, does not contain information on shortcuts. All service information is saved in the file _setup.xml.
2. Supported only MSZIP compression method. If compression will be used for a CAB file, it can be incompatible with PC ActiveSync.
3. It is possible to create custom shortcuts (CabWiz only supports shortcuts to the files included in a CAB). Shortcut target will be exist on device or in CAB file

4. After a CAB file has been edited by CABIntergr, it will clearly not be identical to the initial one.
5. CABIntegr not sign files.

Content

Using CABIntegr

When you start CABIntegr, the main screen show you some options. To select a option, use your stylus or navigation keys.

When you select a option/utility, you will usually see a work window containing a list in the upper part, and an information window in the bottom. If a utility needs to be initialized and/or should examine your device, a status window is displayed before the work window opens. The work list displays a list of items, which have been discovered on your device by the utility; you can perform operation upon these items using the context menus or Action menu. Some utilities let you perform operations upon the complete list of items, or upon several items at once. In this case, each item has a check-box to select it. Delete menu item is applied to the current list item only, even if several items are checked. To delete checked items, use Delete checked command in the menu. Almost every utility has Action->Find and Action->Save list commands. Action->Save list saves the list of all work window items in a text file; this may be useful both for regular usage, and when you want to consult the support service. The information window usually shows the information for the selected item. Please read the information thoroughly. The list can contain several columns. You can sort the list by any column. To choose the sorting mode, you can tap on the column's header, or press a numeric key corresponding to the needed column's number. The information window items can be highlighted with color. You can use the context menu options to copy the information window content to the clipboard, or save it to file. This menu can be activated by pressing Action hard button when the information window has the input focus.

Content

Settings/Items properties dialog

This dialog comprises a screen divided into 3 parts. The lower part contains tabs for different settings sections. In the middle, there is a list of settings related to the selected settings section. The upper part lets you modify the setting selected in the list; it can also contain additional information about this setting. To select a section, use stylus or Left and Right navigation keys. To select a setting, use stylus or Up/Down navigation keys. Pressing Action gets you to the upper part of the screen where you can modify the setting. To return from the setting edit mode, tap the settings list, or press Left navigation arrow. You can use Right navigation key to open the advanced editing mode, similarly to pressing [...] button. To exit the

setting edit mode, select Action->OK or press OK button.

Content

CAB Files

This utility simplifies working with .CAB files (CAB file usually contains Windows Mobile application and its installation information). All CAB files which were found on the device are displayed. To make the search faster, you can exclude some folders by specifying them in the settings.
The application name, vendor name and installation folder name is displayed for each CAB file, if this information is available. There are files which have a CAB-file-like internal structure, but a different filename extension and a different purpose. For example, these are themes and CPF configuration files. Such files can be processed by the utility with some limitations, if this is enabled in the settings. For generic CAB archives created on PCs, unpack operation is available.
What you can do with found files:

1. Create a project for further modification.
2. Delete or move the files.
3. Open a CAB file and explore its contents (files, registry entries, shortcuts. In the explorer mode, use "Action" menu to choose which items to display. Please note that CAB file may not contain files, registry entries or shortcuts. In this case, an empty list will be displayed; this is not an error.
4. Unpack a CAB file. CAB files which are used to install programs on devices, contain files with special names in 8.3 format, and the information about the actual file names.You can extract these files with their actual names, or with the shortened 8.3 names. The files are extracted to the root of the folder you select. If the CAB file contains files with the same name and different paths, only one of them will be extracted.
5. Extract registry keys to REG file
6. Install

Content

Installer

With **Create Installer** or **CAB Files-Action-Add to Installer** menu items you can create one executable file ("Installer"). This executable include one or more files and can install or execute it after start. You can add to Installer:

- CAB file (will be installed)
- REG file (will be imported to registry)
- EXE file (will be executed)
- TXT file (will show with OK,CANCEL question)
- Script (SKSC) file (will be executed)
- any folder(s) and/or file(s) (will be extracted with path)

Action menu commands:

- "Create Installer" - select file name and make Installer.
- "Add command" - select command/file. "Installer" will execute/install/extract this file.
- "Delete" - removes the item.

Controls the sequence of extraction/execution:

- "Up" - move item up.
- "Down" - move item down.

You can save/load configuration by "Profile" command

Contents

Scripts

'Script' is a set of commands stored in a file with the extension *.sksc. This allows to perform several actions during Installer work.

Script sample:

```
#yesno(Stop installation?) #rmr(2) #rmt(10000) show message box with question
#iftrue(yes) if user answer "yes" then go to "yes" label
#goto(end) otherwise go to "end" label
#label(yes) "yes" label definition
#msg(You cancel installation process!) show message box
#stop() stop Installer
#label(end)"end" label definition
```

You can use third party scripting software. For it as first Installer action setup this software:
A) add scripting software CAB file for installation
OR
B) add scripting software EXE file for extraction and add for import REG file with all software settings.
After this you can add to Installer third party scripts (use "Execute custom file command").

Content

Commands

'script' is a set of commands stored in a file with the extension *.sksc.
This file have these format:
#*command 1*(*parameter1;...;parameterN*) [#sleep(*millisecond*)] [other options]
#*command 2*(*parameter1;...;parameterN*) [#sleep(*millisecond*)] [other options]
... #*command N*(*parameter1;...;parameterN*) [#sleep(*millisecond*)] [other options]

Script commands:
#stop (stop Installer execution)
#skip (skip next Installer item)

#r(The name of the file to be launched) #p(parameters), as example #r(\windows\pword.exe) #p(mytext.txt)
#sr (soft reset)
#pwr (power off)
#scr (screen off)
#scn (screen oò)
#kill (kill process, example #kill(pword.exe;fexplorer.exe;))
#hangup hangup
#soundvol set sound volume, from 0 (system) to 15, 16 - mute
#phonevol phone sound volume, from 0 to 5
#script (start script, example #script(script file))
#0 (do nothing)
#theme (change Today theme, if parameter not exist then will be used next theme with filesystem order, , example #theme(\windows\mytheme.tsk))
#wlanon (WLAN on)
#wlanoff (WLAN off)
#bton (BT on)
#btoff (BT off)
#gsmon (GSM on), #gsmon) [#p(PIN)]
#gsmoff (GSM off)
#radioon (all on)
#radiooff (all off)
#findwindow (this command is searching for a window by these criteria (set in the Parameters box of this command): main window title; main window class; child window title, child window class,without parameters find active window
#activatewindow if #findwindow success this command activate obtained window
#getwndtext if #findwindow success this command copy to variable title of obtained window, #getwndtext(4) - will copy title to Var4
#getwndclass if #findwindow success this command copy to variable class name of obtained window, #getwndclass(4) - will copy class name to Var4
#getwndexe if #findwindow success this command copy to variable exe name of obtained window, #getwndexe(4) - will copy executable name to Var4
#postmessage if #findwindow success this command can send message to window , #postmessage(0xMessageCode,0xwParam,0xlParam)
#copyfile copy file #copyfile(source path\filename;destination path\[filename])
#copyfile(My Documents*.*;\Storage Card\My Documents\)
#copyfile(My Documents*.txt;\Storage Card\My Documents\)
#copyfile(My Documents\list.txt;\Storage Card\My Documents\)
#copyfile(My Documents\list.txt;\Storage Card\My Documents\copyoflist.txt)

#move move file or folder #move(source path\filename;destination path\filename) or move

folder #move(source path\folder;destination path\folder)
#move(My Documents*.*;\Storage Card\My Documents\)
#move(My Documents\My Folder;\Storage Card\My Documents\My Folder)
#move(My Documents\list.txt;\Storage Card\My Documents\copyoflist.txt)
#delete delete file or folder #copyfile(file)
#delete(My Documents*.bak)
#delete(\Test Folder)
#led on/off LED, #led(0|1|2)
#tap (this command performs a tap on the screen with X;Y[;X2;Y2] coordinates, if coordinates have not been set, the coordinates of the upper left corner of the window found with the #findwindow command are used). If X2 and Y2 coordinates is present - command emulate tap on X;Y axis and move stylus to X2;Y2 axis.

sample: #tap(30;32) #tap(7;32;30;32)
#tapah (this command performs a tap-and-hold on the screen with X;Y coordinates, if coordinates have not been set, the coordinates of the upper left corner of the window found with the #findwindow command are used)
#msg show message, #msg(Variable A = #A#) #d(12)
#writestr write text to new file, #writestr(filename.txt) #d(Variable A = #A#)
#appendstr append text to file, #appendstr(filename.txt) #d(Variable A = #A#)
#gototoday activate today
#refresh refresh today
#yesno #p(Message text) #d(font size) #rmr(0|1|2 - default answer,0-YES-TRUE,1-NO-FALSE,2-UNKNOWN) #rmt(time limit for show this message,0 - no limit), show message, if user answer YES then sysvar == TRUE, by pressed "action" button or after time (#rmt) will be selected answer from #rmr
#yesno(Are you sure?) #rmr(1) #rmt(10000) - show question, if user not answer - after 10 seconds will be selected "No"
#okcancel see #yesno
#gettext #p(Text) #d(font size) #rmr(0|1|2 - default answer,0-YES-TRUE,1-NO-FALSE,2-UNKNOWN) #rmt(time limit for show this message,0 - no limit) Wait user input. If user enter text and tap OK - sysvar == TRUE and script put text to variable Var0 , if user tap Cancel - sysvar == FALSE. by pressed "action" button or after time (#rmt) will be selected answer from #rmr
#setval set string value to VARx variable, #setval(3) #d(New value for Var3)
#rgdelval delete registry value, #rgdelval(HKLM|HKCU|HKCR;path;value name)
#rgdelkey delete registry key, #rgdelkey(HKLM|HKCU|HKCR;path;key name)
#rgset set registry value, #rgset(HKLM|HKCU|HKCR;path;value name;DWORD|SZ|MZ;value data)
#rgget get registry value, #rgget(HKLM|HKCU|HKCR;path;value name;DWORD|SZ|MZ;variable name)
#regflush flush registry from ram to storage
#getsoundvol get sound volume, #getsoundvol(variable name)
#getphonevol get phone sound volume, #getphonevol(variable name)
#getbattery get battery level, #getbattery(variable name)
#sipon show IM panel, use #chksip for check IM state
#sipoff hide IM panel, use #chksip for check IM state
#getsip get current SIP name to string variable, #getsip(MyVar)
#setsip make SIP "active", r(#setsip(@MyVar|name of SIP)

#url2file download a file, r(#url2file(URL;path on PDA)
#url2file(http://s-k-tools.com/cabintegr/cabintegr.zip;\My Documents\cabintegr.zip)

#filesize get filesize, #filesize(file name;variable name)

#crc32 calculate CRC32 for file, #crc32(file name;variable name)
#getdate get current date , #getdate(date format;string variable number)
#gettime get current time , #gettime(time format;string variable number)
#getdate(dd/MM/yyy;5)
#gettime(hh:mm;6)
#msg(Now @var5 @var6) #d(9)

#rotate - screen rotate command,
#rotate(|0|90|180|270|4|5|6)
empty parameter - rotate next
0|90|180|270 - rotate to
4 - rotate to portrait
5 - rotate to landscape
6 - toggle portrait/landscape

#brightness
#brightness(value;BATTERY|AC)
#brightness(value) - for battery power
#brightness(value;battery) - for battery power
#brightness(value;AC) - for AC power
value - very device dependent
this function also very device dependent, we not guarantee any result

#getmute get "mute" state, #getmute(numeric variable name), very device dependent
#setmute set "mute" state, #setmute(ON|OFF|1|0), very device dependent
#sendkey (to send a keystroke) The following macros can be used in the line:
Enter #R
Tab #T
Backspace #B
Delete #D
Left #<
Right #>
Up #^
Down #!
Home #H
End #N
Escape #E
CNTRL + X #CX, X = any character
Shift ON #S
Shift OFF #s
Hardware buttons (only 15 buttons possible) #A1-#AF
Soft keys (only 2 buttons possible) #K1-#K2
#[any string] - copy "any string" to the clipboard.

#startsound play sound in bacground with rules defined in "script". #startsound(sound file;script)
script (like phone script) commands:
a = activate device
cN = set volume to N in percentage max volume
fN = flash notification LED for N seconds
p = play ringtone. Note that this will play the ringtone all the way through before continuing with the next code.
r = repeat. Note that this should be the last code in your Script string, if used at all.
vN = vibrate for N seconds

wN = wait for N seconds. Note that the device will wait this long before continuing with the next code.
#stopsound stop sound started only from current script.
#stopallsounds stop sound started by any other script(s).

sample (play and vibrate 20 seconds):
#startsound(\Windows\Windows Default.wav;av1pr)
#stopsound() #sleep(20000)

#callbubble call(show) system bubble
#callbubble(command)
command can be:
RADIO,VOLUME,CLOCK,NOTIFICATIONS,DATACONNECTION,CONNECTION

#clipget get text from clipboard #clipget(variable)
#clipset put text to clipboard #clipset(text)
#getpxlclr get color of point of screen #getpxlclr(X;Y;variable)
#speakerphone enable/disable speakerphone mode #speakerphone(ON|1|off|0)

With all command can be used the parameter #sleep(millisecond), this parameter make timeout BEFORE command will run.

back

Script execution sequence control

Commmand #LABEL defines a label. Label is a script line associated with a certain name. This name always makes it possible to find the line. When two identical labels exist, the one which is closer to the beginning of the script is always found.
#LABEL(label name)

Command #GOTO causes a control transfer to the script line defined by a certain label.
#GOTO(label name)

Command #EVAL allow you to set/change a variables.
example:
#eval(a=1)
#label(start)
#msg("A" value = #A#)
#eval(a=a+1)
#chkcondition(a!=4)
#IFTRUE(start)

The following check-up commands can be used in a script:

radio subsystems status	#chkwlan, #chkbt, #chkgsm
screen state (on or off)	#chkscreen
screen orientation (true - landscape, false - portrait, square screen - unknown state)	#chklandscape
files or folders existence	#chkfile, #chkfolder
ActiveSync connection status	#chkasconnection
String include caller ID? (true for yes, false for no, unknown if callerid not defined) #chkcid(+11233456789;9876543;1111111)	#chkcid
Caller ID include string? (true for yes, false for not, unknown if callerid not defined) #chkincid(9876543)	#chkincid
Device power on AC line? (true for yes, false for not, unknown if unknown state)	#chkac
SIP status	#chksip
String1 include any substring from String2? (result - TRUE) Or String1 is substring of String2? (result FALSE) Othervise - result UNKNOWN String2 can include separator ";" #chkstrings(String1) #d(String2) #chkstrings(12345) #d(1234;aaaaaa;bbbbbbb) - result TRUE #chkstrings(aaaaaaaaaa) #d(1234;aaaaaa;bbbbbbb) - result TRUE #chkstrings(aaa) #d(1234;aaaaaa;bbbbbbb) - result FALSE #chkstrings(ccccc) #d(1234;aaaaaa;bbbbbbb) - result UNKNOWN	#chkstrings

You can use #chkcondition for check variables:
#chkcondition(a==10)
You can use #chkconnection for check connection:
#chkconnection(`USB)
You can use #chkprocess for check process:
#chkprocess(tmail.exe)

After these commands have been executed, a special internal variable (for script his name is SYSVAR and possible this: #eval(myvar=sysvar))
takes one of the 3 status: TRUE (on, active, existing), FALSE (off, non-existing), UNKNOWN (not possible to define a status). This variable keeps its status until the following working (not management control one) command is executed. Commands which do not change their status TRUE-FALSE always return the variable into UNKNOWN.

Window search command #findwindow sets this variable value as well depending upon search results.
Files-operation commands like #copyfile, #move, #delete sets this variable value as well depending upon operation results.

Commands #IFTRUE, #IFFALSE, #IFUNKNOWN, #IF cause a control transfer to a label depending upon a status of the internal variable.
#IFTRUE(label name)
#IFFALSE(label name)
#IFUNKNOWN(label name)
#IF(label name for TRUE;label name for FALSE;label name for UNKNOWN)

back

Variables and pseudovariables (macros)

Macros (pseudovariables) is a record like @Name which is replaced with some value if a script is executed. A script may contain the following macros: @time, @date, @var0, ..., @var9 and other - user defined.

You can use variables in scripts. There are two types of variables - numeric and line (string). Numeric variables can contain only numbers. They may have arbitrary names. Usage of up to 50 different numeric variables is allowed.
To work with numeric variables, use the #EVAL command.
to set a value: #eval(a=1)
to change a variable value: #eval(a=a+1)
a variable value can be modified into a character string to appear in a message
#msg("A" value = #A#)
for comparing variables #chkcondition is used
Simple comparison sample
#chkcondition(a!=4)
#IFTRUE(start)
comparison of the current date and the specified date
#eval(data=@Date(yyyyMMdd)) #chkcondition(data>=20060701)

There can only be 50 string variables (macros) altogether.
To access those, @VarName macro are used.
Exist 10 system variables with names @Var0 - @Var9.
Every variable may contain a string of up to 256 symbols.
Variables can be set using command #setvar - (#setvar(variable name|x) #d(text)) (x from 0 to 9) and/or using command #rgget.

back

Installer execution sequence control

Command **#stop()** will stop Installer execution.

Command **#skip()** will skip next Installer item.

You can use this predefined numeric variables:
screenw - screen width
screenx - screen height
osmajor - OS version major
osminor - OS version minor
osbuild - OS version build

You can use this predefined string variables:
aku - AKU
oem - oem device name
platform - device platform
next - next Installer item name

Sample:
You have two cabs: one for devices with QVGA screen and one for devices with VGA screen. You need install only one for right screen resolution. You need create two scripts:
checkvga.sksc
#chkcondition(screenw>=480 & screenh>=480)
#IFTRUE(end)
#skip()
#label(end)
and
checkqvga.sksc
#chkcondition(screenw<480 & screenh<480)
#IFTRUE(end)
#skip()
#label(end)

Now create Installer with this items:
checkvga.sksc
VGA.CAB
checkqvga.sksc
QVGA.CAB

back

Work with strings (text)

To compare variables or strings, use the #chkstrings command
#chkstrings(@var5) #d(123456;aaaaaa;bbbbbbb)

For get string variable length use #length command:
#length(@variable name|text;numeric variable name)

For get substring position use #pos command:
#pos(substring;string;numeric variable name)

For extract substring from string use #substr command:

#substr(string;position;length;result string variable name)

back

Working with ZIP archives

You can a perform basic operations with zip-archives

To create a new, empty archive
#zcreate(archive name) #d(password)
A password is optional.
If there existed a file with the same name - it will be deleted.

To open an existing archive
#zopen(archive name) #d(password)
A password is optional.

To close
#zclose()
closes a new or open archive.

In one script, you can only work with one archive at any point of time.
A working cycle is only limited with commands between #zopen/#zcreate and #zclose.
For creating new archives you can use the command
#zadd(\path\mask)
to add files into an archive.
After an archive is closed (#zclose) you will not be able to modify its contents with script, use other programs for this.
To define how to add files, the following commands are used
#zusepath(ON|off)
this command defines if it is required to save the full path in the name of the file to be added
#zsetrecursion(ON|off)
this command defines if it is required to search in subfolders during a file search.

For existing archives the command
#zextract(\path\mask)
extracts files into the folder which had been set earlier with the command
#zsetextractpath(extract path)
The command
#zusepath(ON|off)
defines if it is required to extract files according to the full path in its name

#zsetnmode(0|1|STD|UTF8)
defines mode for save/restore file names in zip archives

back

REG to CAB

To convert from a REG file, the corresponding menu item has to be selected in the main menu. The ultimate result of this operation is a CAB file which is making changes in the registry while being installed.
Selections to be made:
- whether the resulting CAB file initiates a soft reset after its installation or not;
- REG file name;
- CAB file name;

Content

Project edit window

Edit window contains a list of original parts (files, registry values, shortcuts) of the project. For each project, properties editing feature can be available. There is also a feature to edit the properties for the whole project, add new elements, and delete the existing elements. For new projects, adding elements is only accessible after editing the project properties.

Content

CAB Provisioning file

The Cab Provisioning (.cpf) files used for provisioning and configuring Windows Mobile devices. A .cpf file contains only the _setup.xml file that provides device configuration instructions. CABIntegr can create .cpf files. For it: Open "CPF" menu item, select existing provisioning XML file and enter name for new .cpf file.

Content

Project

Allows to view the existing projects, delete them and switch to the editing mode.

Content

Registration

If you have purchased CABIntegr in online shops, you have received a 5-digit unlock code. Please make sure that Owner Name in your Pocket PC is identical to the one you entered when purchasing the software. Also, if you have some national symbols in your Owner Name, make sure that the corresponding region is selected in your Regional settings. Run CABIntegr and enter your code in the Tools->Registration window. In all other cases you have to send the Windows\CABIntegrid.txt file to the support (support@s-k-tools.com). Please send it as an attached file, not in a mail body. The support will send you an 5-digit unlock code. The procedure for entering it is the same as above.

Content

Support

If you have any suggestions or problems with our software, please feel free to contact us: support@s-k-tools.com

Content

License Agreement

The License Agreement for using CABIntegr (hereinafter referred to as "Software") below is in English language only.

You may:
========
(i) use the Software on any single computer; (ii) use the Software on a second computer as long as the first and second computers are not used simultaneously; and (iii) make a copy of the Software for archival purposes, provided any copy contains all the original Software's proprietary notices. The Software is in "use" when it is loaded into RAM or installed on any storage device(s); (iv) use the Software for personal use only.

You may not:
============
(i) modify, translate, reverse engineer, decompile, disassemble, create derivative works based on the Software, or any portion thereof, or Documentation; or attempt to increase the functionality of the Software in any manner; (ii) copy the Software (except for back-up purposes) or Documentation; (iii) rent, lease, or otherwise transfer rights to the Software or Documentation; or (iv) remove any proprietary notices or labels on or in the Software or Documentation; (v) give your registration information to third parties.

SOFTWARE.
If you receive your first copy of the Software electronically, and a second copy on media, the

second copy may be used for archival purposes only. This license does not grant you any rights to any enhancement or update. TITLE. Title, ownership rights, and intellectual property rights in and to the Software and Documentation belong to the authour of the Software and are protected by the copyright laws of the Russian Federation and international copyright treaties.

LIMITED WARRANTY.
The author guarantees the Software performance within the author's description.
CUSTOMER REMINDERS.
The buyer has the right of receiving newer versions of the Software with corrected errors.

NO OTHER WARRANTIES, EXCEPT AS EXPRESSLY PROVIDED IN THE LIMITED WARRANTY SECTION ABOVE, THE SOFTWARE IS PROVIDED TO THE END USER "AS IS" WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESSED OR LIMITED, INCLUDING, BUT NOT LIMITED TO, WARRANTIES OF NON- INFRINGEMENT, MERCHANTABILITY, AND/OR FITNESS FOR A PARTICULAR PURPOSE. THE ENTIRE RISK OF THE QUALITY AND PERFORMANCE OF THE SOFTWARE IS WITH YOU.

LIMITATION OF LIABILITY.
UNDER NO CIRCUMSTANCES AND UNDER NO LEGAL THEORY, TORT, CONTRACT, OR OTHERWISE, SHALL SK OR ITS SUPPLIERS OR RESELLERS BE LIABLE TO YOU OR ANY OTHER PERSON FOR ANY DAMAGES, INCLUDING WITHOUT LIMITATION INDIRECT, SPECIAL, INCIDENTAL, OR CONSEQUENTIAL DAMAGES OF ANY CHARACTER INCLUDING, WITHOUT LIMITATION, DAMAGES FOR LOSS OF GOODWILL, WORK STOPPAGE, COMPUTER FAILURE OR MALFUNCTION, OR ANY AND ALL OTHER COMMERCIAL DAMAGES OR LOSSES, EVEN IF SK SHALL HAVE BEEN INFORMED OF THE POSSIBILITY OF SUCH DAMAGES, OR FOR ANY CLAIM BY ANY OTHER PARTY. SK SHALL NOT BE LIABLE FOR ANY DAMAGES UNDER THIS AGREEMENT.

TERM AND TERMINATION.
This Agreement will terminate automatically if you fail to comply with the limitations described above. On termination, you must (i) discontinue your use of the Software and (ii) permanently erase or destroy all copies of the Software and Documentation.

Content