



sow - v20 Ada Library User Manual



Sowebio SARL
15, rue du Temple
17310 – St Pierre d'Oléron – France

Capital 15 000 EUR – SIRET 844 060 046 00019 – RCS La Rochelle – APE 6201Z – TVA FR00844060046

sow - v20 Ada Library User Manual

www.soweb.io
contact@soweb.io



CC-by-nc-sa: Attribution + Noncommercial + ShareAlike

ed. 94 of 2023-01-10
page 1 of 115

Ed.	Release	Comments	
1	20210324	Initial release	sr
8	20210402	First review	sr
9	20210404	New Shell_Execute procedure	sr
15	20210412	Refactoring and extend API	sr
23	20210419	Change Humanist 521 BT font to Airbus cockpit free font designed by Intactile ¹	sr
27	20210606	Updates about AIDE 2.14, many enhancements and typos fixed	sr
34	20210804	Add Get_Build function and extend procedure Raise_Exception	sr
38	20211012	Fix typos, add numerous functions, mainly in Vst package	sr
39	20211214	Add Cursor_Off/On & Duration_Stamp_Seconds functions	sr
42	20211220	SQLite integration	sr
46	20220129	SQLite high level integration, add Replace_Pattern, Field_Display and more	sr
48	20220202	Begin Net API with Ip_Check, Mount, Send_Command, Send_File, Unmount	sr
51	20220215	Add some API in Fls and TIO	sr
59	20220306	Many enhancements and bug fixes in most packages	sr
62	20220423	Add an enhance API in Sys	sr
66	20220511	Improve system package management	sr
71	20220625	Add API in Sys	sr
77	20220722	Many enhancements and bug fixes in most packages	sr
93	20230110	Many enhancements and bug fixes in most packages	sr
94			

¹ <https://b612-font.com> under Open Font License, replaced the Humanist 521 BT licensed by Monotype.

❑ Authors

Stéphane Rivière [Number Six] - stef@genesix.org [CTO Sowebio]

Some documentation parts of Sql API are borrowed from low level SQLite driver sources written by Dmitry Kazakov - <http://www.dmitry-kazakov.de>, which is probably one of the most clever SQLite Ada binding ever created.

❑ Manual

Stéphane Rivière [Number Six] - stef@genesix.org [CTO Sowebio]

The “Excuse me I’m French” speech - The main author of this manual is a Frenchman with basic English skills. Frenchmen are essentially famous as frog eaters². They have recently discovered that others ~~forms of communication~~ languages are widely used on earth. So, as a frog eater, I’ve tried to write some stuff in this foreign dialect loosely known here under the name of English. However, it’s a well known fact that frogs don’t really speak English. So your help is welcome to correct this bloody manual, for the sake of the wildebeests, and penguins too.

❑ Syntax notation

Inside a command line:

- A parameter between brackets [] is optional;
- Two parameters separated by | are mutually exclusives.

An important notice:

➤ This is an important notice !

❑ Edition

1 94 - 2023-01-10

²We could be famous as designers of the Concorde, Ariane rockets, Airbus planes or even Ada computer language but, definitely, Frenchmen have to wear beret with bread baguette under their arm to go eating frogs in a smokey tavern. That’s le :]

<https://this-page-intentionally-left-blank.org>



Contents

Introduction.....	13
1 About v20.....	13
2 About the Ada Community.....	13
2.1 Inspiration, ideas, help and more.....	13
3 v20 history.....	14
Getting started.....	15
1 v20 Distribution.....	15
1.1 Directories.....	15
1.2 Key files.....	15
2 Get an Ada compiler.....	15
3 Get v20.....	15
4 v20 build.....	16
v20 Constants.....	17
1 ANSI colors for console.....	17
2 Control characters.....	17
3 Delimiter characters.....	17
4 Flag files.....	18
5 Redirection.....	18
v20 API.....	19
1 Introduction.....	19
1.1 Concepts.....	19
1.2 Conventions.....	19
1.3 Usage.....	19
2 v20.....	19
2.1 Get_Build.....	20
2.2 Get_Log_Dir.....	20
2.3 Get_Tmp_Dir.....	20
2.4 Get_Version.....	21
2.5 Raise_Exception.....	21
3 Cfg - Configuration files.....	22
3.1 Close.....	22
3.2 Comment.....	22
3.3 Delete.....	22
3.4 Get.....	22
3.5 Open.....	23
3.6 Set.....	23
4 Fls - Files.....	23
4.1 Backup_File.....	23
4.2 Copy_File.....	24

4.3	Create_Directory_Tree.....	24
4.4	Delete_Directory_Tree.....	24
4.5	Delete_File.....	25
4.6	Delete_Lines.....	25
4.7	Download_File.....	26
4.8	Exists.....	26
4.9	Extract_Directory.....	27
4.10	Extract_Name.....	27
4.11	File_Size.....	27
4.12	Get_Directory.....	28
4.13	Is_Root_Directory.....	28
4.14	Move_File.....	28
4.15	Rename.....	29
4.16	Search_Lines.....	29
4.17	Set_Directory.....	29
5	Log - Logging.....	30
5.1	Dbg.....	30
5.2	Err.....	30
5.3	Get_Debug.....	30
5.4	Get_Dir.....	30
5.5	Line.....	31
5.6	Msg.....	31
5.7	Set_Debug.....	31
5.8	Set_Dir.....	32
5.9	Set_Display.....	32
5.10	Set_Disk.....	32
5.11	Set_Header.....	32
5.12	Set_Task.....	33
5.13	Title.....	33
6	Net - Network.....	33
6.1	Command.....	33
6.2	Copy_File.....	34
6.3	Delete_Directory_Tree.....	35
6.4	Delete_File.....	35
6.5	Directory_Exists.....	36
6.6	File_Exists.....	36
6.7	Get_Network_From_Ip.....	36
6.8	Is_Ip_Ok.....	37
6.9	Is_Ping_Ok.....	37
6.10	Is_Root_Directory.....	38
6.11	Is_Ssh_Ok.....	38
6.12	Mount.....	38
6.13	Mount_Remote.....	39
6.14	Set_Exception.....	39

6.15	Set_Hostname.....	40
6.16	Set_Key.....	40
6.17	Set_Message.....	40
6.18	Set_Output.....	41
6.19	Unmount.....	41
6.20	Unmount_Remote.....	42
7	Prg - Program.....	42
7.1	Command.....	42
7.2	Current_Time_Seconds.....	42
7.3	Duration_Stamp.....	43
7.4	Duration_Stamp_Seconds.....	43
7.5	Duration_Stamp_Time.....	43
7.6	Generate_Password.....	44
7.7	Get_Version.....	44
7.8	Get_Version_Major.....	44
7.9	Get_Version_Minor.....	45
7.10	Is_User_Not_Root.....	45
7.11	Name.....	45
7.12	Path.....	46
7.13	Set_Exit_Status.....	46
7.14	Set_Version.....	46
7.15	Start_Dir.....	46
7.16	Start_Time.....	47
7.17	Time_Stamp.....	47
8	Sql - SQLite.....	47
8.1	Bind.....	48
8.2	Column_Integer.....	48
8.3	Column_Text.....	49
8.4	Column_Count.....	49
8.5	Column_Exists.....	50
8.6	Column_Type.....	50
8.7	Delete.....	51
8.8	Error.....	51
8.9	Error_Display.....	52
8.10	Exec.....	52
8.11	Get_Config.....	53
8.12	Get_Version.....	53
8.13	Index_Exists.....	53
8.14	Insert.....	54
8.15	Last_Insert_RowID.....	55
8.16	Last_RowID.....	55
8.17	Open.....	56
8.18	Prepare.....	56
8.19	Row_Count.....	57

8.20	Read.....	57
8.21	Reset.....	58
8.22	Schema_Load.....	58
8.23	Schema_Need_Update.....	59
8.24	Schema_Update.....	60
8.25	Search.....	60
8.26	Set_Config.....	61
8.27	Step.....	61
8.28	Table_Exists.....	62
8.29	Update.....	63
9	Sys - System.....	63
9.1	Command_Path.....	63
9.2	Get_Alloc_Ada.....	64
9.3	Get_Alloc_All.....	64
9.4	Get_Env.....	64
9.5	Get_Home.....	65
9.6	Get_Memory_Dump.....	65
9.7	Get_System_Name.....	66
9.8	Get_System_Version.....	67
9.9	Install_Packages.....	67
9.10	Is_Command.....	68
9.11	Is_Package.....	68
9.12	Purge_Packages.....	68
9.13	Reset_Memory_Monitor.....	69
9.14	Set_Env.....	69
9.15	Set_Memory_Monitor.....	69
9.16	Shell_Execute.....	70
10	Tio - Text console.....	71
10.1	Animated_Delay.....	71
10.2	Ansi.....	72
10.3	Beep.....	72
10.4	Clear_Screen.....	72
10.5	Confirm_Twice.....	72
10.6	Cursor_Line_Backward.....	73
10.7	Cursor_Line_Erase.....	73
10.8	Cursor_Line_Forward.....	73
10.9	Cursor_Line_Move.....	74
10.10	Cursor_Off.....	74
10.11	Cursor_On.....	74
10.12	Cursor_Restore.....	74
10.13	Cursor_Save.....	75
10.14	Line.....	75
10.15	Get_Immediate.....	75
10.16	Get_Password.....	76

10.17	Pause.....	76
10.18	Put.....	76
10.19	Put_Line.....	77
11	Tio - Text files.....	77
11.1	Append.....	77
11.2	Close.....	77
11.3	Create.....	78
11.4	End_Of_Line.....	78
11.5	End_Of_File.....	79
11.6	Flush.....	79
11.7	Get.....	79
11.8	Get_Line.....	80
11.9	Is_Open.....	80
11.10	Line.....	80
11.11	Open_Conf.....	81
11.12	Open_Read.....	81
11.13	Put.....	82
11.14	Put_Line.....	82
11.15	Read_File.....	82
11.16	Reset.....	83
11.17	Write_File.....	83
12	Vst - VStrings.....	83
12.1	Char_Count.....	83
12.2	Element.....	84
12.3	Empty.....	84
12.4	Ends_With.....	84
12.5	Field_* functions guidelines.....	85
12.6	Field_By_Index.....	85
12.7	Field_By_Name.....	85
12.8	Field_Count.....	86
12.9	Field_Included.....	86
12.10	Field_Display.....	87
12.11	Field_Search.....	87
12.12	Head.....	88
12.13	Index.....	88
12.14	Index_backward.....	88
12.15	Length.....	89
12.16	Replace_Char.....	89
12.17	Replace_Pattern.....	90
12.18	Slice.....	90
12.19	Starts_With.....	90
12.20	Stript_Accents.....	91
12.21	Stript_Chars.....	91
12.22	Tail.....	92

12.23	Tail_After_Match.....	92
12.24	To_Lower.....	93
12.25	To_Upper.....	93
12.26	Trim_Both.....	94
12.27	Trim_Left.....	94
12.28	Trim_Right.....	94
12.29	Trim_Slashes.....	95
12.30	+.....	95
12.31	*.....	95
12.32	&.....	96
12.33	=.....	96
12.34	<.....	96
12.35	<=.....	97
12.36	>.....	97
12.37	>=.....	97
13	Vst - Types conversion and tests.....	98
13.1	Ascii_Value_To_Hex.....	98
13.2	Is_Numeric.....	98
13.3	To_Hex.....	98
13.4	To_Integer.....	99
13.5	To_String.....	99
13.6	To_Val.....	99
13.7	To_VString.....	100
	v20 architecture.....	101
1	Introduction.....	101
2	Requirements.....	101
3	Coding guidelines.....	101
3.1	General.....	101
3.2	Messages.....	101
3.3	Naming.....	101
4	Design.....	102
4.1	Types.....	102
4.2	Packages.....	102
4.3	Functions.....	103
	FAQ.....	105
1	Conventional exit codes.....	105
2	Converting reminder.....	106
2.1	Converting Integer to String with Character'Val and Integer'Image.....	106
2.2	Converting a character to its ASCII value.....	106
2.3	Converting VString from and to Long_Integer.....	106
3	How to prepare SQLite to v20 integration.....	106
3.1	Simple Components.....	106
3.2	SQLite.....	107
	Programs examples.....	109

1	test.adb.....	109
Appendices.....		111
1	Copyrights & credits.....	111
1.1	Library Licence.....	111
1.2	Manual license.....	111
1.3	v20 Packages copyrights & credits.....	111
2	To-do list.....	111
2.1	v20.Sql.....	111
2.2	v20.Tio.....	112
2.3	Doc.....	112
3	Quality control.....	112
4	Release check list.....	112
5	Issues.....	113
5.1	Compiler bug reporting.....	113

<https://this-page-intentionally-left-blank.org>



Introduction

1 About v20

v20 is a Ada library for Linux service and console programs, primary designed to be used in Genesix, a cluster manager for High Availability virtual instances on GNU/Linux Debian/Xen servers.

However, v20 is a general purpose library, KISS³ oriented and very efficient to create any command line program. AIDE uses v20 too.

v20 is also a modular library with components designed to work together. Naming and conventions are consistent. Currently, v20 is composed of ten packages in charge of unbounded strings, program and OS functions, console and text files, network, SQLite handling, logging and configuration files handling. At least five other packages are planned, related to databases and web APIs, without being limited to these aspects only.

2 About the Ada Community



At first, all our warmly thanks to the Ada Community, definitely one of the best.

2.1 Inspiration, ideas, help and more

AdaCore Ada compiler - <https://www.adacore.com/community>

Daniel Feneuille - df- <http://d.feneuille.free.fr>

Gautier de Montmollin - gdm - <https://github.com/zertovitch>

Jean-Pierre Rosen - jpr - <https://adalog.fr>

Pascal Pignard - pp - <https://github.com/Blady-Com>

Rolf Ebert - re - <https://github.com/RREE>

Special thanks to Ada gurus Daniel Feneuille, Gautier de Montmollin and Jean-Pierre Rosen. The chapter heading quotes are extracted from Murphy's Law and

³ Keep It Simple, Stupid - https://en.wikipedia.org/wiki/KISS_principle - In memory of <http://www.nason-line.org/publications/biographical-memoirs/memoir-pdfs/johnson-clarence.pdf> the genius father of titanium Blackbirds.

other reasons why things go wrong - A. Bloch. They come from <https://www.ada-log.fr> site created by Jean-Pierre Rosen.

3 v20 history

We own the copyrights for v89, v90, v93, v95, v04 and v20. Some work in v20 is derived from theses.

Ver.	Langages	Proc.	Système	Context	Copyright	Users
v87	Clipper	i386	MsDos	ST Formation	Proprietary	CEA-DAM CEA EDF
v89	Clipper/C/Asm	i386	MsDos	Atlansys	Proprietary	ETDE SAMU EDF
v90	Clipper/C/Asm	i386	MsDos	Atlansys	Proprietary	Military NGO EDF
v93	C++	i386	Windows	Atlansys	Proprietary	Research
v95	Delphi	i386	Windows	Astriane	Proprietary	Military NGO
v96	Asm	st62xx	Embedded	MRT	Proprietary	Military Civilian
v97	Asm	pic17c44	Embedded	MRT	Proprietary	Military Civilian
v04	Ada	i386	Windows	AIDE v1	GMGPL	Education
v20	Ada	All	Linux	AIDE v2	GPL v3	General Purpose

Getting started

One can write neatly in any language, including C. One can write badly in any language, including Ada. But Ada is the only language where it is more tedious to write badly than neatly.

Jean-Pierre Rosen



1 v20 Distribution

1.1 Directories

v20 comes with some inner directories:

Packages	Description
bin	test binary place, with dontdelete.me test file for trailing comments preservation
doc	place of sow - v20 Ada Library User Manual.pdf and others documentation files
doc-generated	API doc generated by GNATStudio with GNATDoc
obj/debug obj/fast obj/small	build directories
src	sources of v20
src/sys	specials system files as s-memory.adb, the GNATColl memory monitory hook
src-tests	sources of v20 tests programs

1.2 Key files

Key files are located in the main directory.

v20.gpr project file for building v20 with GNAT

2 Get an Ada compiler

Just use AIDE: <https://github.com/sowebio/aide-bin>

3 Get v20

You can get v20 at <https://github.com/sowebio/v20>

sow - v20 Ada Library User Manual.odt

4 v20 build

□ Compilation

Assuming you wish to install v20 under <your path> with a GNAT compiler already installed, do the following from a command line interpreter. Open a terminal:

```
user@system: cd <your path>
user@system: git clone https://github.com/sowebio/v20
user@system: cd v20
user@system: gprbuild -P v20
user@system: cd bin
user@system: ./test
```

v20 Constants

Investment in C programs reliability will increase up to exceed the probable cost of errors or until someone insists on recoding everything in Ada.
Gilb's laws synthesis



1 ANSI colors for console

This constants conforms to ISO 6429 standard :

```
CONSOLE_COLOR_GREEN : constant String := ESC & "[1; 32m";  
CONSOLE_COLOR_RED   : constant String := ESC & "[1; 31m";  
CONSOLE_COLOR_YELLOW: constant String := ESC & "[1; 33m";  
CONSOLE_COLOR_RESET : constant String := ESC & "[0m";
```

2 Control characters

Common control characters :

```
HT   : constant String := Character'Val[9] & " "; -- 09d 09h Tab  
LF   : constant String := Character'Val[10] & " "; -- 10d 0Ah Line Feed  
CR   : constant String := Character'Val[13] & " "; -- 13d 0Dh Carriage return  
ESC  : constant String := Character'Val[27] & " "; -- 27d 1Bh Escape  
DQ   : constant String := Character'Val[34] & " "; -- 34d 22h Double quote  
CRLF : constant String := CR & LF;
```

3 Delimiter characters

V20 conventional delimiter characters :

```
ND : constant String := "~"; -- 126d 7Eh Name/value delimiter  
CD : constant String := "^"; -- 94d 5Eh Column delimiter  
RD : constant String := "\"; -- 92d 5Ch Row delimiter  
VD : constant String := ","; -- 44d 2Ch Virgule [comma] delimiter  
SD : constant String := ":"; -- 58d 3Ah String delimiter  
SP : constant String := " "; -- 32d 20h Space
```

⇒ Some of these delimiters are heavily used in the v20.Vst.Field_* functions.

4 Flag files

Useful names for testing mounts or install completed :

```
ACCESS_OK : constant String := "access_ok_dont_delete_this_file";  
INSTALL_OK : constant String := "install_ok_dont_delete_this_file";
```

5 Redirection

Output redirections for standard and error outputs.

```
STD_OUT_REDIRECT : constant String := " 1>/dev/null";  
ERR_OUT_REDIRECT : constant String := " 2>/dev/null";  
STD_ERR_OUT_REDIRECT : constant String := " 2>/dev/null 1>/dev/null";
```

v20 API

There are 10 types of people in the world: those who understand binary and those who don't.

Anonymous



1 Introduction

1.1 Concepts

The developer is a writer. The writer's courtesy is clarity;

Clarity and ease of use are prioritized over speed and efficiency.

The performance of a compiled language such as Ada as well as the hardware capabilities of current systems justify these choices.

On a simple loop, let's recall that if HAC [Ada subset interpreter] is [among others] 7 times faster than Bash, HAC itself is 300 times slower than Ada.

1.2 Conventions

To ease developers:

➤ All strings constants and function only returns VString typed.

➤ All strings parameters accept both String and VString types.

1.3 Usage

The HAC runtime is located in the `./v20/src` directory.

Use `./v20/v20.gpr` as a stub for your own projects.

Use `./v20/src-tests/test.adb` as an template to integrate the appropriate v20 with and use clauses.

2 v20

Base package.

2.1 Get_Build

- Description

Returns the formatted build date stamp as: "build YYYY-mm-dd hh:mm:ss".

- Usage

function Get_Build return VString

- Example

```
Log.Msg ["Date stamp build: " & v20.Get_Build];  
build 2021-08-04 14:36:27
```

2.2 Get_Log_Dir

- Description

Returns the log directory.

- Usage

function Get_Log_Dir return VString

- Example

```
Log.Msg ["v20.Get_Log_Dir];  
/var/log/
```

2.3 Get_Tmp_Dir

- Description

Returns the temporary files directory.

- Usage

function Get_Tmp_Dir return VString

- Example

```
Log.Msg ["v20.Get_Tmp_Dir];  
/tmp/
```

2.4 Get_Version

- Description

Returns the Library name and formatted version: "<space>v.minor.major".

- Usage

function Get_Version return VString

- Example

```
Log.Msg ["Library version: " & v20.Get_Version];  
v20 v0.6
```

2.5 Raise_Exception

- Description

Raise an exception for reporting test and <program_Name.err> file creation.

In addition to the usual trace, a v20 exception give some extra information like: exception time, program uptime, program & library names & versions, start & home directories and Ada and all languages memory allocation, current & maximum [peak] values.

- Usage

procedure Raise_Exception

- Example

```
Raise_Exception;  
  
-----  
Exception time      : 20210402 160834  
Program uptime     : 0h00m00s  
Program name & version: test v0.2  
Library name & version: v20 v0.1  
Start directory    : /home/sr/Seafire/Sowebio/informatique/dev/ada/prj/v20/  
bin  
Home directory     : /home/sr  
Ada memory allocations: Ada Cur: [ 2272 ] Max: [ 201912 ]  
All memory allocations: All Cur: [ 3465216 ] Max: [ 3465216 ]  
  
raised V20.RAISE_EXCEPTION.V20_EXCEPTION_TEST : v20.adb: 47  
[./test]  
V20.Raise_Exception at v20.adb: 47  
Test at test.adb: 311  
Main at b__test.adb: 375  
0x475937 __libc_start_main at ???  
0x4053c8 _start at ???  
  
-----
```

3 Cfg - Configuration files

3.1 Close

- Description

Close Cfg file. For sanity only as each setting is instantly flushed to disk.

- Usage

procedure Close

- Example

<<<TODO>>>

3.2 Comment

- Description

Insert a comment Text after the last line of the config file.

- Usage

procedure Comment [Text : String]

- Example

<<<TODO>>>

3.3 Delete

- Description

Delete parameter in section. If no other parameter in this section, delete section too. Avoid reserved chars [] = # inside parameters.

- Usage

procedure Delete [Section : String; Parameter : String]

- Example

<<<TODO>>>

3.4 Get

- Description

Return parameter in section or empty string if not found. Avoid reserved chars [] = # inside parameters.

- Usage

function Get [Section : String; Parameter : String] return VString

- Example

<<<TODO>>>

3.5 Open

- Description

Open and load if exist a configuration file. Create blank if non existent. Default configuration file name is “program name” followed by “.cnf” extension and created in the program start directory.

- Usage

```
function Open [Cfg_File_Read_In : String := ""] return Boolean
```

- Example

<<<TODO>>>

3.6 Set

- Description

Create or replace an existing parameter in a section. If this latter does not exist, also creating it. New setting is persistent even program quits unexpectedly after. Avoid reserved chars [] = # inside parameters. If reserved chars are passed, the procedure does nothing. An optional trailing comment can also be added.

- Usage

```
procedure Set [Section : String; Parameter : String; Value : String; Comment : String := ""]
```

- Example

<<<TODO>>>

4 Fls - Files

4.1 Backup_File

- Description

Rename file with .bak.n suffix. Iterate n=0..9 searching a free n bak file. If n is free then write .bak.n, if n=9, delete .bak.0

- Usage

```
procedure Backup_File [File_To_Backup : String];  
procedure Backup_File [File_To_Backup : VString];
```

- Example

<<<TODO>>>

4.2 Copy_File

- Description

Copy a Source_Name file to a Target_Name file destination.

Copy_Form is “preserve=all_attributes,mode=overwrite” [full attributes preservation and overwrite file if exists].

- Usage

```
procedure Copy_File [Source_Name, Target_Name : String]
procedure Copy_File [Source_Name, Target_Name : VString]
procedure Copy_File [Source_Name : VString; Target_Name : String]
procedure Copy_File [Source_Name : String; Target_Name : VString]
```

- Example

<<<TODO>>>

4.3 Create_Directory_Tree

- Description

Create a directory tree Dir_Tree. Each non-existent directory named by Dir_Tree is created [possibly including other intermediate directories]. Return False if operation is unsuccessful [i.e. if base directory tree is inconsistent or still don't exist after the creating attempt]. Return True if directory tree already exists or has just been created.

Extra inner slashes are processed i.e. a directory like /home/sr/opt/ytr.lkj/////kjghgh will be valid. and will create, from /home/sr/opt :

- Directory ytr.lkj
- And then inner directory kjghgh

- Usage

```
function Create_Directory_Tree [Dir_Tree : String] return Boolean
function Create_Directory_Tree [Dir_Tree : VString] return Boolean
```

- Example

<<<TODO>>>

4.4 Delete_Directory_Tree

- Description

Delete a directory tree Dir_Tree. The directory and all of its contents [possibly including other directories] are deleted. Return True if Dir_Tree is successfully

deleted or was already deleted. Return False if operation is unsuccessful [i.e. if base directory tree was non existent or still exists after the deleting attempt].

Dir_Tree must be fully qualified, i.e. starting with a slash [/].

This function prevents deletion of the following root directories: bin, boot, dev, etc, home, lib, lib32, lib64, libx32, lost+found, media, mnt, opt, proc, root, run, sbin, srv, sys, tmp, usr, var. Pay close attention, you can't delete /etc but you are allowed to delete /etc/network !

/! With programs ran with root rights, this routine should be used with infinite caution.

/! This function uses Ada.Directories.Delete_Tree, which raises an exception if the directory tree to delete contains a **broken** symbolic link [a file like any other]. This latter is seen as **non-existent** and, when the parent directory is deleted, an exception occurs : raised ADA.IO_EXCEPTIONS.USE_ERROR : directory tree rooted at <directory tree> could not be deleted [because **not empty**]. Funny, but not so much. Pure C code problem in Ada RTS. Stacked C calls in russian puppet mode until a logical problem arises.

- Usage

```
function Delete_Directory_Tree [Dir_Tree : String] return Boolean
function Delete_Directory_Tree [Dir_Tree : VString] return Boolean
procedure Delete_Directory_Tree [Dir_Tree : String]
procedure Delete_Directory_Tree [Dir_Tree : VString]
```

- Example

<<<TODO>>>

4.5 Delete_File

- Description

Delete a Name file only if this latter exists. No exception will be raised if the file to delete does not exists.

- Usage

```
procedure Delete_File [Name : String]
procedure Delete_File [Name : VString]
```

- Example

<<<TODO>>>

4.6 Delete_Lines

- Description

Search and remove file lines matching Pattern in File_Name.

- Usage

```
procedure Delete_Lines [File_Name, Pattern : String]
procedure Delete_Lines [File_Name, Pattern : VString]
procedure Delete_Lines [File_Name : String; Pattern : VString]
procedure Delete_Lines [File_Name : VString; Pattern : String]
```

- Example

<<<TODO>>>

4.7 Download_File

- Description

Download a file from Url to Dfile. Do nothing if Dfile already exists with its size equals Dsize. Name is purely informational and used to named file in text messages.

Return True is Dfile present at the right size, False otherwise.

- Usage

```
function Download_File [Url : VString;
                        Dfile : VString;
                        Name : VString;
                        Dsize : Integer := 0] return Boolean;
```

- Example

<<<TODO>>>

4.8 Exists

- Description

Returns True if file or directory Name exists.

- Usage

```
function Exists [Name : String] return Boolean
function Exists [Name : VString] return Boolean
```

- Example

```
if Exists [HAC_Dir & "/hac"] then
  Put_Line ["HAC installation is done : "];
end if;
```

4.9 Extract_Directory

- Description

Returns directory from Name.

- Usage

```
function Extract_Directory [Name : String] return VString  
function Extract_Directory [Name : VString] return VString
```

- Example

```
Put_Line [Extract_Directory ["/etc/ssh/sshd_config"]] then  
/etc/ssh
```

4.10 Extract_Name

- Description

Returns filename from Name.

- Usage

```
function Extract_Filename [Name : String] return VString  
function Extract_Filename [Name : VString] return VString
```

- Example

```
Put_Line [Extract_Filename ["/etc/ssh/sshd_config"]] then  
sshd_config
```

4.11 File_Size

- Description

Return size of Name file.

- Usage

```
function File_Size [Name : String] return Integer  
function File_Size [Name : VString] return Integer
```

- Example

<<<TODO>>>

4.12 Get_Directory

- Description

Returns current directory.

- Usage

```
function Current_Directory return String  
function Current_Directory return VString
```

- Example

<<<TODO>>>

4.13 Is_Root_Directory

- Description

This function checks the following root directories: bin, boot, dev, etc, home, lib, lib32, lib64, libx32, lost+found, media, mnt, opt, proc, root, run, sbin, srv, sys, tmp, usr, var. Returns True if Dir_Tree is a root directory.

Dir_Tree must be fully qualified, ie starting with a slash [/].

- Usage

```
function Is_Root_Directory [Dir_Tree : String] return Boolean  
function Is_Root_Directory [Dir_Tree : VString] return Boolean
```

- Example

```
Put_Line [Is_Root_Directory ["/etc"]];  
True  
  
Put_Line [Is_Root_Directory ["/etc/network"]];  
False
```

4.14 Move_File

- Description

Move a Source_Name file to a Target_Name file destination. Copy_Form is “pre-serve=all_attributes,mode=overwrite” [full attributes preservation and overwrite file if exists].

- Usage

```
procedure Move_File [Source_Name, Target_Name : String]  
procedure Move_File [Source_Name, Target_Name : VString]  
procedure Move_File [Source_Name : VString; Target_Name : String]  
procedure Move_File [Source_Name : String; Target_Name : VString]
```

- Example

<<<TODO>>>

4.15 Rename

- Description

Rename an Old_Name file or directory to a New_Name file or directory. If exists a file New_File, it will be overwritten.

- Usage

```
procedure Rename {Old_Name, New_Name : String}
procedure Rename {Old_Name, New_Name : VString}
procedure Rename {Old_Name : VString; New_Name : String}
procedure Rename {Old_Name : String; New_Name : VString}
```

- Example

<<<TODO>>>

4.16 Search_Lines

- Description

Search at least a line matching Pattern in File_Name and return true if found.

- Usage

```
function Search_Lines {File_Name, Pattern : String} return Boolean
function Search_Lines {File_Name, Pattern : VString} return Boolean
function Search_Lines {File_Name : String; Pattern : VString} return Boolean
function Search_Lines {File_Name : VString; Pattern : String} return Boolean
```

- Example

<<<TODO>>>

4.17 Set_Directory

- Description

Change to a directory Directory. Create Directory if this latter does not exist, return False if operation failed.

- Usage

```
function Set_Directory {Directory : String} return Boolean
function Set_Directory {Directory : VString} return Boolean
```

- Example

<<<TODO>>>

5 Log - Logging

5.1 Dbg

- Description

Log a debug message. 45 characters max before truncation with a maximum line length of 79.

- Usage

```
procedure Dbg [Message : in String]
procedure Dbg [Message : in VString]
```

- Example

<<<TODO>>>

5.2 Err

- Description

Log an error message. 45 characters max before truncation with a maximum line length of 79.

- Usage

```
procedure Err [Message : in String]
procedure Err [Message : in VString]
```

- Example

<<<TODO>>>

5.3 Get_Debug

- Description

Return true if debug status is on.

- Usage

```
function Get_Debug return Boolean
```

- Example

<<<TODO>>>

5.4 Get_Dir

- Description

Returns log file directory.

- Usage

function Get_Dir return VString

- Example

<<<TODO>>>

5.5 Line

- Description

Log a blank line.

- Usage

procedure Line

- Example

<<<TODO>>>

5.6 Msg

- Description

Log a message. 45 characters max before truncation with a maximum line length of 79.

- Usage

```
procedure Msg [Message : in Boolean];
procedure Msg [Message : in Integer]
procedure Msg [Message : in Long_Integer]
procedure Msg [Message : in Character]
procedure Msg [Message : in String]
procedure Msg [Message : in VString]
```

- Example

<<<TODO>>>

5.7 Set_Debug

- Description

Set debug messages status on/[off].

- Usage

procedure Set_Debug [Action : Boolean]

- Example

<<<TODO>>>

5.8 Set_Dir

- Description

Set log file directory.

- Usage

```
procedure Set_Dir [Dir_In : String]
procedure Set_Dir [Dir_In : VString]
```

- Example

<<<TODO>>>

5.9 Set_Display

- Description

Log to display on/[off].

- Usage

```
procedure Set_Display [Action : Boolean]
```

- Example

<<<TODO>>>

5.10 Set_Disk

- Description

Log to disk on/[off].

- Usage

```
procedure Set_Disk [Action : Boolean]
```

- Example

<<<TODO>>>

5.11 Set_Header

- Description

Line header on/[off].

- Usage

```
procedure Set_Header [Action : Boolean]
```


- Example

<<<TODO>>>

5.12 Set_Task

- Description

Set new current log task name. 7 characters max before truncation.

- Usage

function Log_Dir return String

- Example

<<<TODO>>>

5.13 Title

- Description

Log a title. 45 characters max before truncation with a maximum line length of 79.

- Usage

```
procedure Title [Message : in String];
procedure Title [Message : in VString];
```

- Example

<<<TODO>>>

6 Net - Network

6.1 Command

- Description

Send remote command to host. Returns True if command successful [remote exit-code equal to 0]. If used, SE_Output returns remote console output.

- Usage

```
function Command [Target : in VString ; Command_In : in VString ; SE_Output : out
VString] return Boolean
function Command [Target : in VString ; Command : in VString] return Boolean
procedure Command [Target : in VString ; Command : in VString]
```

- Exception

Error_Command	Raised when send command error
---------------	--------------------------------

- Example

List files in a directory:

```
Send_Command ["root@i51c1.domain.tld", +"cd /root; ls -l"];
```

```
drwxr-xr-x 7 root      root      4.0K Sep  1 10:45 acme.sh
-rw-r--r-- 1 root      root      3.4K Aug  5 09:28 aide.err
-rw-r--r-- 1 root      root      12K Aug  5 09:53 aide.log
-rw-r--r-- 1 root      root        1 Aug  5 09:28 check.gpr
drwxr-xr-x 2 root      root      4.0K Dec 11 15:02 dmf
-rwxr-xr-x 1 root      root      2.7M Dec 14 11:37 gprbuild
drwxr-xr-x 3 root      root      4.0K Aug  5 09:53 opt
-rw-r--r-- 1 root      root      47M Sep 25 11:37 s015.sql
-rw-r--r-- 1 root      root     134 Aug  7 17:14 test.txt
```

Complex command example [massive URL change in wordpress DB]:

```
Net.Command ["root@i152c1", +"cd /srv/www/adm152.temp_domain.tld/sar ; php
srdb.cli.php -h localhost -n dmf_transfert -u dmf -p " & Pwd_DB_Prod & " -s
https://www.old_domain.tld -r https://www.new_domain.tld"];
```

6.2 Copy_File

- Description

Copy to distant host. Returns True if copy successful.

Options allows extra parameters, like -r for recursive copy.

- Usage

```
function Copy_File [Target : in VString ; File_Tx : in VString; Directory_Rx : in
VString ; Options : in VString := +"" ] return Boolean;
procedure Copy_File [Target : in VString ; File_Tx : in VString; Directory_Rx : in
VString ; Options : in VString := +""];
```

- Exception

Error_Copy	Raised when send file error
------------	-----------------------------

- Example

```
Copy /home/sr/text.txt to root@i51c1.domain.tld/etc/genesix/test.txt
```

```
Net.Copy_File ["root@i51c1.domain.tld", +"/home/sr/text.txt", +"/etc/genesix"];
```

```
Recursive copy: Copy directory and subdirectories content from /etc/genesix/
templates/files-debian-10/usr to n203c1:/etc/xen-tools/skel. This recursive
copy Will transfer ./usr/* to n203c1:/etc/xen-tools/skel/usr/*
```

6.3 Delete_Directory_Tree

- Description

Delete a directory tree `Dir_Tree`. The directory and all of its contents [possibly including other directories] are deleted but adding a '*' at the end of the path preserve the last directory of the path [/one/two/ deletes two but /on/two/* preserve two.

Return True if `Dir_Tree` is successfully deleted or was already deleted. Return False if operation is unsuccessful [i.e. if base directory tree was non existent or still exists after the deleting attempt].

`Dir_Tree` must be fully qualified, ie starting with a slash [/]. This function prevents deletion of the following root directories [see `Is_Root_Directory` for further details]. Pay close attention, you can't delete /etc but you are allowed to delete /etc/network !

Delete a directory tree `Dir_Tree`. The directory and all of its contents [possibly including other directories] are deleted. Return True if `Dir_Tree` is successfully deleted or was already deleted. Return False if operation is unsuccessful [i.e. if base directory tree was non existent or still exists after the deleting attempt].

`Dir_Tree` must be fully qualified, i.e. starting with a slash [/].

This function prevents deletion of the following root directories: bin, boot, dev, etc, home, lib, lib32, lib64, libx32, lost+found, media, mnt, opt, proc, root, run, sbin, srv, sys, tmp, usr, var. Pay close attention, you can't delete /etc but you are allowed to delete /etc/network !

/ With programs ran with root rights, this routine should be used with infinite caution.

- Usage

```
function Delete_Directory_Tree [Target : in VString ; Dir_Tree : VString] return Boolean
```

- Example

<<<TODO>>>

6.4 Delete_File

- Description

Remove `File_To_Delete` in remote host `Target`. Returns True if delete successful.

- Usage

```
function Delete_File [Target : in VString ; File_To_Delete : in VString] return Boolean;
```

```
procedure Delete_File [Target : in VString ; File_To_Delete : in VString];
```

- Example

```
Copy /home/sr/text.txt to root@i51c1.domain.tld/etc/genesix/test.txt  
Net.Delete_File [+"root@i51c1.domain.tld", +"/home/sr/text.txt"];
```

6.5 Directory_Exists

- Description

Returns True if distant directory Name exists.

- Usage

```
function Directory_Exists [Target : in VString, Name : String] return Boolean  
function Directory_Exists [Target : in VString, Name : VString] return Boolean
```

- Example

```
if Directory_Exists [+"host", +"Directory_to_test"] then  
  Put_Line ["Directory_to_test exists"];  
end if;
```

6.6 File_Exists

- Description

Returns True if file Name exists.

- Usage

```
function File_Exists [Target : in VString, Name : String] return Boolean  
function File_Exists [Target : in VString, Name : VString] return Boolean
```

- Example

```
if File_Exists [+"host", +"filename_to_test"] then  
  Put_Line ["filename_to_test exists"];  
end if;
```

6.7 Get_Network_From_Ip

- Description

Returns the network part of a /32 classless IP address.

- Usage

```
function Get_Network_From_Ip [Ip : in String] return VString;
```

```
function Get_Network_From_Ip [Ip : in VString] return VString;
```

- Example

```
Tio.Put_Line [ Get_Network_From_Ip ["120.1.1.1"]];  
120.1.1  
Tio.Put_Line [ Get_Network_From_Ip ["320.1.1.1"]];  
Empty string
```

6.8 Is_Ip_Ok

- Description

IP validation

- Usage

```
function Is_Ip_Ok [IP : in String] return Boolean;  
function Is_Ip_Ok [IP : in VString] return Boolean;
```

- Example

```
Tio.Put_Line [Is_Ip_Ok ["320.1.1.1"]];  
False  
Tio.Put_Line [Is_Ip_Ok ["120.1.1.1"]];  
True
```

6.9 Is_Ping_Ok

- Description

Return true if target answer to a ping.

- Usage

```
function Is_Ping_Ok [Target : in VString] return Boolean;
```

- Example

```
Net.Is_Ping_Ok ["This_host_exists"];  
True  
Net.Is_Ping_Ok ["This_host_don't_exist"];  
False
```

6.10 Is_Root_Directory

- Description

This function checks the following root directories: bin, boot, dev, etc, home, lib, lib32, lib64, libx32, lost+found, media, mnt, opt, proc, root, run,/sbin, srv, sys, tmp, usr, var. Returns True if Dir_Tree is a root directory.

Dir_Tree must be fully qualified, ie starting with a slash [/].

- Usage

function Is_Root_Directory [Dir_Tree : VString] return Boolean

- Example

```
Put_Line [Is_Root_Directory ["/etc"]];
True

Put_Line [Is_Root_Directory ["/etc/network"]];
False
```

6.11 Is_Ssh_Ok

- Description

Return true if target answer to a ping.

- Usage

function Is_Ssh_Ok [Target : in VString] return Boolean;

- Example

```
Net.Is_Ssh_Ok [+"This_host_exists_with_valid_Ssh_Credentials"];
True

Net.Is_Ssh_Ok [+"This_host_exists_without_valid_Ssh_Credentials"];
False
```

6.12 Mount

- Description

Mount a Target as host.

If local admin, automatically create local mount point in
- /mnt/<Target>

If local [non root] user, automatically create local mount point in
- /home/<user>/mnt/<Target>

- Usage

```
procedure Mount [Target : VString];
```

- Example

```
Local admin case: Mount [+"root@i51c1.domain.tld"];

Mounts target root@i51c1.domain.tld: /
to /mnt/root@i51c1.domain.tld

Local <user> case [with home at /home/user]: Mount [+"root@i51c1.domain.tld"];

Mounts target sr@i51c1.domain.tld: /
to /home/<user>/mnt/sr@i51c1.domain.tld
```

6.13 Mount_Remote

- Description

Mount a Mount_Point targetting Target_To_Mount on Remote_Host with options Mount_Options. All mount options are accepted. Returns true if operation is successful.

- Usage

```
function Mount_Remote [Remote_Host : VString ; Target_To_Mount : VString ;
Mount_Point : VString ; Mount_Options : in VString := +"" ] return Boolean
procedure Mount_Remote [Remote_Host : VString ; Target_To_Mount : VString ;
Mount_Point : VString ; Mount_Options : in VString := +""];
```

- Example

```
if Mount_Remote [+"user@remote_host.org", +"/dev/vg/lvm_volume", +"/tmp/mount-
point", +"-o ro"] then
  Tio.Put ["Mount point is mounted read-only"];
end if;
```

6.14 Set_Exception

- Description

Enable Exception processing, which is disabled by default. A call without parameter returns the Exception status [enable or disabled].

- Usage

```
procedure Set_Exception [Set_Unset : Boolean := True]
function Set_Exception return Boolean
```

- Example

```
Private_Key := Sql.Read ["Tbl_Cluster", "Key_Private", "WHERE Number = 1"];
Set_Exception;
```

6.15 Set_Hostname

- Description

Set Hostname for a Target host. Returns true if command ok.

- Usage

function Set_Hostname [Target : VString ; Hostname : VString] return Boolean

- Example

```
Private_Key := Sql.Read ["Tbl_Cluster", "Key_Private", "WHERE Number = 1"];

if Set_Hostname ["i11c1", "i110c1"] then
  Tio.Put ["Hostname is changed"];
end if;
```

6.16 Set_Key

- Description

Set SSH private key used to log in distant hosts with commands like Send_Command and Send_File. Key validity is checked. Returns true if Key is properly set. A call without parameter delete the key previously set.

- Usage

function Set_Key [Key : VString := ""] return Boolean;
procedure Set_Key;

- Example

```
Private_Key := Sql.Read ["Tbl_Cluster", "Key_Private", "WHERE Number = 1"];

if Set_Key [Private_Key] then
  Tio.Put ["Key is set"];
end if;
```

6.17 Set_Message

- Description

Control message output when using commands like Send_Command and Send_File. Default is console message enable. A call without parameter enable message output.

- Usage

```
procedure Set_Message [Msg : Boolean := True];
```

- Example

```
- Disable console message when using commands like Send_Command and Send_File.  
Set_Message [False];
```

6.18 Set_Output

- Description

Control console output when using commands like Send_Command and Send_File. Default is console output enable. A call without parameter enable console output.

- Usage

```
procedure Set_Output [Output : Boolean := True];
```

- Example

```
- Disable console output when using commands like Send_Command and Send_File.  
Set_Output [False];
```

6.19 Unmount

- Description

Unmount a mount point on a remote host.

The local mountpoint directory is deleted.

- Usage

```
procedure Unmount [Target : VString];
```

- Exception

Error_Unmount	Raised when unmount error
---------------	---------------------------

- Example

```
Local admin case: Unmount ["root@i51c1.domain.tld"];  
Unmounts /mnt/root@i51c1.domain.tld  
Local <user> case [home is /home/user]: Unmount ["root@i51c1.domain.tld"];  
Unmounts /home/<user>/mnt/sr@i51c1.domain.tld
```

6.20 Unmount_Remote

- Description

Unmount a Mount_Point on a Remote_Host. Mount_Point is then deleted. Returns true if the whole operation is successful.

- Usage

```
function Unmount_Remote (Remote_Host : VString ; Mount_Point : VString) return Boolean
procedure Unmount_Remote (Remote_Host : VString ; Mount_Point : VString);
```

- Example

```
if Unmount_Remote [+"user@remote_host.org", +"/tmp/mountpoint"] then
  Tio.Put ["Mount point is unmounted"];
end if;
```

7 Prg - Program

7.1 Command

- Description

Constant storing program command [Arg 0].

- Usage

Command : constant VString

- Example

```
Tio_Line [Command];
/home/sr/Seafire/Sowebio/informatique/dev/ada/app/gnx/src/gnx-instance
```

7.2 Current_Time_Seconds

- Description

Returns a duration as seconds since ISO date 19700101. Conforms to Unix time standard. Checked with date +%s. Compliant algorithm until 2070.

Returns a duration in seconds since current time.

- Usage

function Current_Time_Seconds return Natural

- Example

```
Log.Msg ["Current time in seconds: " & To_VString [Current_Time_Seconds]];
1646227335
```

7.3 Duration_Stamp

- Description

Returns a duration as HHhMMmSSs since Time.

- Usage

```
function Duration_Stamp [Time : Ada.Calendar.Time] return VString
```

- Example

```
Log.Msg ["Total execution time: " & Prg.Duration_Stamp [Prg.Start_Time]];

```

7.4 Duration_Stamp_Seconds

- Description

Returns a duration as seconds since Time.

- Usage

```
function Duration_Stamp_Seconds [Time : Ada.Calendar.Time] return Natural
```

- Example

<<<TODO>>>

7.5 Duration_Stamp_Time

- Description

Returns a formatted HHhMMmSSs VString from Time_Seconds.

- Usage

```
function Duration_Stamp_Time [Time_Seconds : Integer] return VString
```

- Example

```
Tio.Put_Line ["Total execution time: " & Prg.Duration_Stamp_Time [1646315044]];
13h35m34s
```

7.6 Generate_Password

- Description

Password generation with 64 charset $[A-Z] + [a-z] + [0-9] + '_' + '-'$
Search space size greater than $1,26 \times 10^{25}$
Space exploration time: 40000 centuries @ 100 billion tests per second.

Command line with standard tools: `< /dev/urandom tr -dc _A-Z-a-z-0-9 | head -c${1:-14};echo;` Generates 14 chars long passwords like: 5fx7_0Fubo-hNa

- Usage

function Generate_Password return VString

- Example

```
Log.Msg [Generate_Password];  
5fx7_0Fubo-hNa
```

7.7 Get_Version

- Description

Returns formatted program version : "<space>v.minor.major".

- Usage

function Get_Version return VString

- Example

```
Log.Msg ["Program version: " & prg.Get_Version];  
Program version: v2.16
```

7.8 Get_Version_Major

- Description

Returns Major version.

- Usage

function Get_Version_Major return Natural;

- Example

```
Log.Msg [prg.Get_Version_Major];
```

7.9 Get_Version_Minor

- Description

Returns Minor version.

- Usage

function Get_Version_Minor return Natural;

- Example

```
Log.Msg [prg, Get_Version_Minor];
```

```
16
```

7.10 Is_User_Not_Root

- Description

Returns true if program user's not root.

- Usage

function Is_User_Not_Root return Boolean

- Example

<<<TODO>>>

7.11 Name

- Description

Return program name.

- Usage

function Name return VString

- Example

```
sr@ro8 ~/Seafire/Sowebio/informatique/github/aide/bin > aide
```

```
aide
```

7.12 Path

- Description

Return program path.

- Usage

function Path return String

- Example

```
sr@ro8 ~/Seafire/Sowebio/informatique/github/aide/bin > aide  
/home/sr/Seafire/Sowebio/informatique/github/aide/bin
```

7.13 Set_Exit_Status

- Description

Set errorlevel return code. Each call is cumulative. Four calls with 1, 2, 4 and 8 set 15 ie msb-00001111-lsb. Can be used everywhere in the program without special call at its end.

Convention : 1 = no or bad command, 128 = runtime exception [8th bit].

- Usage

procedure Set_Exit_Status [Code : Natural]

- Example

<<<TODO>>>

7.14 Set_Version

- Description

Set program version.

- Usage

procedure Set_Version [Major : Natural; Minor : Natural]

- Example

<<<TODO>>>

7.15 Start_Dir

- Description

Constant storing current directory at start.

- Usage

Start_Dir : constant VString

- Example

<<<TODO>>>

7.16 Start_Time

- Description

Constant storing Time at program start.

- Usage

Start_Time : constant Ada.Calendar.Time

- Example

<<<TODO>>>

7.17 Time_Stamp

- Description

Returns current timestamp as YYYYMMDD-HHMMSS

- Usage

function Time_Stamp return VString

- Example

<<<TODO>>>

8 Sql - SQLite

See v20-sql.adb to see full high and low level examples.

A comprehensive "SQLite digest manual" is available to ease SQLite newcomers. See Sowebio Github repository.

- Tech notes

Closing Database and Statement are automatically handling by Finalize procedures, thanks to Dmitry Kazakov low level SQLite driver.

SQLite DB is fully statically linked in projects using V20 [SQLite dynamic extensions are disabled].

8.1 Bind

- Description

Set a parameter of statement.

The parameters to be bound are usually specified as ? in the command text [see Prepare]. Each such parameter has to be bound to a value. The position of a parameter is specified by its index, i.e. by the position of ? in the command text. The first parameter has the position 1.

- Exceptions

Constraint_Error	Command or Parameter is invalid
Data_Error	Data base error
End_Error	Not found [table does not exist]
Status_Error	Access errors
Use_Error	File access related errors

- Usage

```
procedure Bind [Parameter : Positive; Value : Integer];
procedure Bind [Local_Handle_Statement : Statement; Parameter : Positive; Value : Integer];
procedure Bind [Parameter : Positive; Value : VString];
procedure Bind [Local_Handle_Statement : Statement; Parameter : Positive; Value : VString];
```

- Example

```
Key := "key" & Trim_Left[To_VString[Integer' Image[Index]]];
Value := "value"& Trim_Left[To_VString[Integer' Image[Index]]];

Tio.Put_Line ["Insert Key: " & Key & " with value: " & Value];

Sql.Bind [1, Key];
Sql.Bind [2, Value];
```

8.2 Column_Integer

- Description

Return a Integer from a column in the current result row, whatever the column type.

- Usage

```
function Column_Integer [Position : Positive] return Integer;
function Column_Integer [Local_Handle_Statement : Statement; Position : Positive] return Integer;
```


- Exceptions

Constraint_Error

Command is an invalid handle

- Example

```
Tio.Put [Sql.Column {2}];
1234
```

8.3 Column_Text

- Description

Return a VString from a column in the current result row, whatever the column type.

- Usage

```
function Column_Text [Position : Positive] return VString;
function Column_Text [Local_Handle_Statement : Statement; Position : Positive]
return VString;
```

- Exceptions

Constraint_Error

Command is an invalid handle

- Example

```
for Index in 1..Columns loop
  Tio.Put [Sql.Column_Text {Index}];
end loop;
```

8.4 Column_Count

- Description

Get the number of columns in the current result set.

- Usage

```
function Column_Count return Natural;
function Column_Count [Local_Handle_Statement : Statement] return Integer;
```

- Exceptions

Constraint_Error

Command is an invalid handle

- Example

```
Columns := Sql.Column_Count;
```

```
Tio.Put_Line ["Column count: " & To_VString [Integer' Image[Columns]]];
```

8.5 Column_Exists

- Description

Return true if Column_Name exists.

- Usage

```
function Column_Exists [Table_Name : String; Column_Name : String] return Boolean;  
function Column_Exists [Table_Name : VString; Column_Name : VString] return Boolean;
```

- Exceptions

Constraint_Error	Base is an invalid handle
Data_Error	Data base error
End_Error	Not found [table does not exist]
Status_Error	Access error
Use_Error	File open error

- Example

```
Tio.Put ["Column_Exists: "];  
Tio.Put_Line [Column_Exists ["test_table", "Existing_Column"]]; -- Existing  
column  
  
Tio.Put ["Column_Exists: "];  
Tio.Put_Line [Column_Exists ["test_table", "azeazeaze"]]; -- Non existing  
column  
  
...  
  
Column_Exists: True  
Column_Exists: False
```

8.6 Column_Type

- Description

Get a column type in the current result row

- Usage

```
function Column_Type [Position : Positive] return Datatype;  
function Column_Type [Local_Handle_Statement : Statement; Position : Positive]  
return Datatype;
```

- Exceptions

Constraint_Error	Command is an invalid handle
------------------	------------------------------

- Example

```
...
for Index in 1..Columns loop
  Tio.Put [Sql.Column [Index] & " [T" & Trim_Left[To_VString[Sql.Datatype' Image
[Sql.Column_Type [Index]]]] & "]" "];
end loop;
...

Row 1 : key11 [T3] value4 [T3] 11 [T1]
Row 2 : key12 [T3] value4 [T3] 12 [T1]
Row 3 : key13 [T3] value4 [T3] 13 [T1]
Row 4 : key14 [T3] value4 [T3] 14 [T1]
Row 5 : key4 [T3] value4 [T3] 4 [T1]

T1 = Integer
T3 = Text
```

8.7 Delete

- Description

Delete a row in Table_Name specifying a Where_Condition

- Usage

```
procedure Delete [Table_Name : VString ; Where_Condition : VString];
```

- Exceptions

Constraint_Error	Base is an invalid handle
Data_Error	Data base error
End_Error	Not found [table does not exist]
Status_Error	Access error
Use_Error	File open error

- Example

```
-- Delete row for Number = 1234 in table Cluster
Sql.Delete ["Cluster", "Number = 1234"];
```

8.8 Error

- Description

SQLite error codes processing. Returns a numeric code and an Information_Extended string with more comprehensive information.

- Usage

```
function Error [Information : String, Information_Extended : in out VString := +""]  
return Natural;
```

- Example

The program attempts to create a row with an non existent foreign key.

.../...

```
exception
  when Fault : others =>
    Exception_Result := Sql.Error [AE.Exception_Information [Fault],
Exception_Result_Extended];

    Tio.Put_Line ["Exception_Result: " & To_VString [Exception_Result]];
    Tio.Put_Line ["Exception_Result_Extended: " & Exception_Result_Extended];
```

```
Exception_Result: 19
Exception_Result_Extended: A SQL constraint violation occurred
```

8.9 Error_Display

- Description

SQLite status, info and error display. Returns a VString with more comprehensive information than raw Error_Code.

- Usage

```
function Error_Display [Error_Code : Natural] return VString;
```

- Example

```
Tio.Put_Line [Error_Display [19]]
A SQL constraint violation occurred
```

8.10 Exec

- Description

Execute a SQL command when no output is needed. It's a wrapper around Prepare, Step and Finalize, that allows an application to run multiple statements of SQL without having to use a lot of code. Command is UTF-8 encoded.

- Usage

```
procedure Exec [Command : String];
procedure Exec [Command : VString];
```

- Exceptions

Constraint_Error	Base is an invalid handle
Data_Error	Data base error
End_Error	Not found [table does not exist]
Status_Error	Access error

- Example

```
-- Write ahead log transaction mode, safe write to avoid corruption
Sql.Exec {"PRAGMA journal_mode=WAL; PRAGMA synchronous=FULL"};

-- Table setup
Sql.Exec {"DROP TABLE IF EXISTS test_table"};
Sql.Exec {"CREATE TABLE test_table [key TEXT PRIMARY KEY, value TEXT, valuenum
INTEGER]"};
```

8.11 Get_Config

- Description

Get configuration Value from Parameter stored in Config table.

- Usage

```
function Get_Config [Parameter : String] return VString
function Get_Config [Parameter : VString] return VString
```

- Example

```
-- Get parameter's value 'Schema_Version' [previously set to '0.1']
Get_Config ["Schema_Version"];

0.1
```

8.12 Get_Version

- Description

Return SQLite version, with a x.y.z format.

- Usage

```
function Get_Version return VString;
```

- Example

```
Tio.Put_Line [Sql.Get_Version];

sql v3.37.0
```

8.13 Index_Exists

- Description

Return true if Index_Name exists.

- Usage

```
function Index_Exists [Index_Name : VString] return Boolean;
```

- Exceptions

Constraint_Error	Base is an invalid handle
Data_Error	Data base error
End_Error	Not found [table does not exist]
Status_Error	Access error
Use_Error	File open error

- Example

```
Tio.Put["Index_Exists: "];
Tio.Put_Line [Index_Exists {"key"}]; -- Existing index

Tio.Put["Index_Exists: "];
Tio.Put_Line [Column_Exists {"key1"}]; -- Non existing index

...

Index_Exists: True
Index_Exists: False
```

8.14 Insert

- Description

Create a row in Table_Name with Columns_Values.

The special character ^ is used to separate column/value pairs and the special character ~ is used to distinguish the name of a column from its value. See example below.

- Usage

```
procedure Insert [Table_Name : VString; Columns_Values : VString];
```

- Exceptions

Constraint_Error	Base is an invalid handle
Data_Error	Data base error
End_Error	Not found [table does not exist]
Status_Error	Access error
Use_Error	File open error

- Example

```
-- Fill Number with 1234 and Domain with genesix.org in table Cluster

Sql.Insert [{"Cluster", "Number~1234" & "^" & "Domain~genesix.org"}]
```

8.15 Last_Insert_RowID

- Description

The function usually returns the rowid of the most recent successful INSERT into a rowid table or virtual table. Inserts into WITHOUT ROWID tables are not recorded. If no successful INSERTs into rowid tables have ever occurred on the database, then the function returns zero .

Each entry in most SQLite tables [except for WITHOUT ROWID tables] has a unique 64-bit signed integer key called the "rowid". The rowid is always available as an undeclared column named ROWID, OID, or _ROWID_ as long as those names are not also used by explicitly declared columns. If the table has a column of type INTEGER PRIMARY KEY then that column is another alias for the rowid [this text from https://www.sqlite.org/c3ref/last_insert_rowid.html].

- Usage

function Last_Insert_RowID return Integer_64;

- Exceptions

Constraint_Error	Base is an invalid handle
Data_Error	Data base error
End_Error	Not found [table does not exist]
Status_Error	Access error
Use_Error	File open error

- Example

```
-- This is the 14th insert [see test.adb in v20 scr-tests directory]

Sql.Exec  ["INSERT INTO test_table [Key, Value, vnum] VALUES
['key14', 'value4', 14]; "];
Tio.Put  ["Last_Insert_Row_ID: "];
Tio.Put_Line [Sql.Last_Insert_RowID];

14

-- Other example [Number INTEGER PRIMARY KEY UNIQUE]

Sql.Exec ["INSERT INTO Cluster [Number,Domain,] VALUES [1234,'genesix.org'"];
Tio.Put_Line ["Insert_RowID: " & Trim_Left [To_VString [Integer [Sql.Last_In-
sert_RowID]]]];

Insert_RowID: 1234
```

8.16 Last_RowID

- Description

Returns last existing RowID in Table_Name.

- Usage

function Row_Count [Table_Name : String; Option : String := "*"] return Natural;

- Example

```
Tio.Put_Line [Row_Count ["Table_test"]];
12
```

8.17 Open

- Description

Open a database.

- Usage

procedure Open [Database_File_Name : VString];

- Exceptions

Data_Error	Data base error
Use_Error	File open error

- Example

```
Sql.Open ["sqlite_high_level_test.db"];
```

8.18 Prepare

- Description

Prepare a SQL command when an output is needed. Command is UTF-8 encoded.

- Usage

procedure Prepare [Statement_To_Prepere : VString];
function Prepare [Statement_To_Prepere : VString] return Statement;

- Exceptions

Constraint_Error	Base is an invalid handle
Data_Error	Data base error
End_Error	Not found [table does not exist]
Status_Error	Access error
Use_Error	File open error

- Example

```
Sql.Prepare ["DELETE FROM test_table WHERE key=?"];
```

```

for Index in 1..Count / 3 loop
  Sql.Exec ["BEGIN TRANSACTION; "];
  Key := +"key" & Trim_Left[To_VString[Integer' Image[Index]]];
  Tio.Put_Line ["Delete row with Key: " & Key];
  Sql.Bind [1, Key];
  Sql.Step;
  Sql.Reset;
  Sql.Exec ["COMMIT; "];
end loop;

```

8.19 Row_Count

- Description

Returns counted rows in Table_Name with Options.

Option:

- '*' is all rows, included null-ed
- 'Column name' counts not null-ed rows
- 'DISTINCT Column name' counts not null-ed and distinct rows

- Usage

function Row_Count [Table_Name : String; Option : String := "*"] return Natural;

- Example

```

Tio.Put_Line [Row_Count ["Table_test"]];
12

```

8.20 Read

- Description

Returns an extraction from Table_Name with comma delimited Columns and standard SQL Condition [like WHERE, ORDER BY, LIMIT].

The extraction is formatted with standard v20 CD constant as Column delimiter and RD constant as Row delimiter.

- Usage

function Read [Table_Name : VString; Columns : VString; Condition : VString := ""]
return VString;

- Exceptions

Constraint_Error	Base is an invalid handle
Data_Error	Data base error
End_Error	Not found [table does not exist]
Status_Error	Access error

- Example

```

Sql.Read ["Cluster", +"Number, Domain"];
Sql.Read ["Cluster", +"Number, Domain", +"WHERE Number = 1234"];

-- Used in combination with Field_Display

Field_Display [Sql.Read ["Cluster", +"Number, Domain"], CD, RD, "Cluster num-
ber, Domain name"];

Cluster number   Domain name
-----
1                domain1
2                domain2
3                domain3
4                domain4
1234             genesix2.org

```

8.21 Reset

- Description

Complete SQL command execution, make it ready to execute again.

- Usage

```

procedure Reset;
procedure Reset [Local_Handle_Statement : Statement];

```

- Exceptions

Constraint_Error Command is an invalid handle

- Example

```

Sql.Reset;

```

8.22 Schema_Load

- Description

Load a schema. Commands will be executed by Schema_Update in code source order

- Usage

```

procedure Schema_Load [Command : in Schema_Command := Null_Command ;
Name : in String := "" ; Attribute : in String := ""];

```

- Example

```

Sql.Schema_Load [Sql.Table_Name,      "Cluster"];
Sql.Schema_Load [Sql.Column_Name,     "Number",  "INTEGER"];
Sql.Schema_Load [Sql.Column_Constraint, "Number",  "UNIQUE"];
Sql.Schema_Load [Sql.Table_Constraint, "Number",  "PRIMARY KEY"];
Sql.Schema_Load [Sql.Column_Name,     "Domain",  "TEXT"];
Sql.Schema_Load [Sql.Column_Name,     "Email",   "TEXT"];
Sql.Schema_Load [Sql.Column_Name,     "Manager", "INTEGER"];
Sql.Schema_Load [Sql.Index_Name,       "Idx",     "Number"];
Sql.Schema_Load [Sql.Index_Constraint, "Idx",     "UNIQUE"];

```

8.23 Schema_Need_Update

- Description

Open or Create Database_FullName, with a Major and Minor minimum schema version. If schema version is upper than the version stored in Database_FullName, returns True as a database schema update is needed.

The Schema_Load and Schema_Update is therefore only launched when necessary.

To update the database schema in table Cluster with a new column named 'Bidule', simply :

- Increment Sql.Schema_Need_Update ["Sqlite_Update_Test", 0, 1] to 0,2
- Add at the right place Sql.Schema_Load [Sql.Column_Name, "Bidule", "TEXT"];

See examples below.

- Usage

```
function Schema_Need_Update [Database_FullName : String ; Major : Natural; Minor : Natural] return Boolean;
```

```
function Schema_Need_Update [Database_FullName : VString ; Major : Natural; Minor : Natural] return Boolean;
```

- Examples

- Before schema update

```

if Sql.Schema_Need_Update ["Sqlite_Update_Test", 0, 1] then

    Sql.Schema_Load [Sql.Table_Name,      "Cluster"];
    Sql.Schema_Load [Sql.Column_Name,     "Number",  "INTEGER"];
    Sql.Schema_Load [Sql.Column_Constraint, "Number",  "UNIQUE"];
    Sql.Schema_Load [Sql.Table_Constraint, "Number",  "PRIMARY KEY"];
    Sql.Schema_Load [Sql.Column_Name,     "Domain",  "TEXT"];
    Sql.Schema_Load [Sql.Column_Name,     "Email",   "TEXT"];
    Sql.Schema_Load [Sql.Column_Name,     "Manager", "INTEGER"];
    Sql.Schema_Load [Sql.Index_Name,       "Idx",     "Number"];
    Sql.Schema_Load [Sql.Index_Constraint, "Idx",     "UNIQUE"];

    Sql.Schema_Update;

end if;

```

- After schema update

```

if Sql.Schema_Need_Update ["Sqlite_Update_Test", 0, 2] then

    Sql.Schema_Load [Sql.Table_Name,          "Cluster"];
    Sql.Schema_Load [Sql.Column_Name,         "Number", "INTEGER"];
    Sql.Schema_Load [Sql.Column_Constraint,   "Number", "UNIQUE"];
    Sql.Schema_Load [Sql.Table_Constraint,    "Number", "PRIMARY KEY"];
    Sql.Schema_Load [Sql.Column_Name,         "Domain", "TEXT"];
    Sql.Schema_Load [Sql.Column_Name,         "Email", "TEXT"];
    Sql.Schema_Load [Sql.Column_Name,         "Bidule", "TEXT"];
    Sql.Schema_Load [Sql.Column_Name,         "Manager", "INTEGER"];
    Sql.Schema_Load [Sql.Index_Name,          "Idx", "Number"];
    Sql.Schema_Load [Sql.Index_Constraint,     "Idx", "UNIQUE"];

    Sql.Schema_Update;

end if;

```

8.24 Schema_Update

- Description

Create and delete tables, table constraints, columns, columns constraints, index, index constraints on database schema after loading schema by Schema_Load.

<<<TODO>>> : implement delete and backup DB before update or delete.

- Usage

procedure Schema_Update;

- Example

```

if Sql.Schema_Need_Update ["Sqlite_Update_Test", 0, 1] then

    Sql.Schema_Load [Sql.Table_Name,          "Cluster"];
    Sql.Schema_Load [Sql.Column_Name,         "Number", "INTEGER"];
    Sql.Schema_Load [Sql.Column_Constraint,   "Number", "UNIQUE"];
    Sql.Schema_Load [Sql.Table_Constraint,    "Number", "PRIMARY KEY"];
    Sql.Schema_Load [Sql.Column_Name,         "Domain", "TEXT"];
    Sql.Schema_Load [Sql.Column_Name,         "Email", "TEXT"];
    Sql.Schema_Load [Sql.Column_Name,         "Manager", "INTEGER"];
    Sql.Schema_Load [Sql.Index_Name,          "Idx", "Number"];
    Sql.Schema_Load [Sql.Index_Constraint,     "Idx", "UNIQUE"];

    Sql.Schema_Update;

end if;

```

8.25 Search

- Description

Return True if Condition verified

- Usage

function Search [Table_Name : VString; Condition : VString] return Boolean;

- Example

```
if Sql.Search ["Cluster",+"WHERE Number = 1234"] then
  Tio.Put_Line ["Search 'Number = 1234': Found"];
end if;
if not Sql.Search ["Cluster",+"WHERE Number = 9999"] then
  Tio.Put_Line ["Search 'Number = 9999': Not found"];
end if;

Search 'Number = 1234': Found
Search 'Number = 9999': Not found

if Sql.Search ["Cluster",+"WHERE Login = 'sr'"] then
  Tio.Put_Line ["Search 'Login = sr': Found"];
end if;

Search 'Login = sr': Found
```

8.26 Set_Config

- Description

Store configuration Parameter and Value to Config table.

- Usage

```
procedure Set_Config (Parameter : String ; Value : String);
procedure Set_Config (Parameter : VString ; Value : VString);
```

- Example

```
-- Set '0.1' value in parameter 'Schema_Version'
Set_Config ["Schema_Version", "0.1"];
```

8.27 Step

- Description

Execute prepared command

When the result is False, the Command execution has been completed. In this case the next operation should be Reset. When the result is True there is a row of data produced by the command. The next operation can be Step to get another row or else Reset to reset the statement. After calling Reset, the parameters can be rebound before another execution of the parameter is initiated by doing Step.

- Usage

```
procedure Step;
procedure Step [Local_Handle_Statement : Statement];
function Step return Boolean;
function Step [Local_Handle_Statement : Statement] return Boolean;
```

- Exceptions

Constraint_Error	Command is an invalid handle
Data_Error	Data base error
End_Error	Not found [table does not exist]
Status_Error	Access error
Use_Error	File open error

- Example

```

Sql.Prepare ["INSERT INTO test_table VALUES (?, ?, ?)"];

for Index in 1..Count loop
  Sql.Exec ["BEGIN TRANSACTION; "];
  -- Primary key so keys must be unique
  Key := "key" & Trim_Left[To_VString[Integer' Image[Index]]];
  Value := "value"& Trim_Left[To_VString[Integer' Image[Index]]];
  Tio.Put_Line ["Insert Key: " & Key & " with value: " & Value];
  Sql.Bind [1, Key];
  Sql.Bind [2, Value];
  Sql.Bind [3, Index];
  Sql.Step;
  Sql.Reset;
  Sql.Exec ["COMMIT; "];
end loop;

```

8.28 Table_Exists

- Description

Return true if Table_Name exists.

- Usage

```

function Table_Exists [Table_Name : String] return Boolean;
function Table_Exists [Table_Name : VString] return Boolean;

```

- Exceptions

Constraint_Error	Command is an invalid handle
Data_Error	Data base error
Status_Error	Access error
Use_Error	File open error

- Example

```

Tio.Put["Table_Exists: "];
Tio.Put_Line [Table_Exists ["test_table"]]; -- Existing table

Tio.Put["Table_Exists: "];
Tio.Put_Line [Table_Exists ["test_table1"]]; -- Non existing table

...

Table_Exists: True

```

Table_Exists: False

8.29 Update

- Description

Update a row in Table_Name with Columns_Values specifying a Where_Condition.

The special character ^ [or constant ND as Name/value delimiter] is used to separate column/value pairs and the special character ~ [or constant CD as Column delimiter] is used to distinguish the name of a column from its value. See example below.

- Usage

```
procedure Update [Table_Name : VString; Columns_Values : VString;
Where_Condition : VString];
```

- Exceptions

Constraint_Error	Base is an invalid handle
Data_Error	Data base error
End_Error	Not found [table does not exist]
Status_Error	Access error
Use_Error	File open error

- Example

```
-- Update Domain column with genesix2.org value for Number = 1234 in table
Cluster
Sql.Update ["Cluster", "Domain~genesix2.org", "Number = 1234"];
```

9 Sys - System

9.1 Command_Path

- Description

Return full qualified command path or an empty string if not found.

- Usage

```
function Command_Path [Command_Name : String] return VString;
function Command_Path [Command_Name : VString] return VString;
```

- Example

```
Command_Path [ "mc" ];  
/usr/bin/mc
```

9.2 Get_Alloc_Ada

- Description

Return current and max allocations done from Ada excluding others languages. Format of returned string : Ada Cur: [868] Max: [1600].

- Usage

function Get_Alloc_Ada return String;

- Example

```
Prg.Get_Alloc_Ada;  
Ada Cur: [ 868 ] Max: [ 1600 ]
```

9.3 Get_Alloc_All

- Description

Return current and max allocations done from all languages including Ada. Format of returned string: Ada Cur: [868] Max: [1600]. This uses system calls to find out the program's resident size [RSS] information, both the peak and the current size.

- Usage

function Get_Alloc_All return String;

- Example

```
Prg.Get_Alloc_All;  
All Cur: [ 2514944 ] Max: [ 2514944 ]
```

9.4 Get_Env

- Description

Returns VString value of VString or String environment variable Name

- Usage

function Get_Env [Name : String] return VString

function Get_Env [Name : VString] return VString

- Example

<<<TODO>>>

9.5 Get_Home

- Description

Returns HOME path without trailing slash.

- Usage

function Get_Home return VString

- Example

```
Get_Home - for user 'sr'
"/home/sr"
```

9.6 Get_Memory_Dump

- Description

Dump information about memory usage. Size is the number of the biggest memory users we want to show. Report indicates which sorting order is used, depending of the following options:

- Prg.All_Reports;
- Prg.Memory_Usage;
- Prg.Allocations_Count;
- Prg.Sort_Total_Allocs;
- Prg.Marked_Blocks;

➤ You must activate memory monitor with Set_Memory_Monitor before using this function.

- Usage

```
procedure Get_Memory_Dump [Size : Positive; Report_View : Report :=
Memory_Usage]
```

```
Prg.Get_Memory_Dump [1];
```

- Example

Displaying all report options :

```
Prg.Get_Memory_Dump [1];
```

```
Traceback elements allocated: 2480
```

Validity elements allocated: 1

Ada Allocs: 60608 bytes in 1258 chunks
Ada Free: 60008 bytes in 1248 chunks
Ada Current watermark: 600 in 10 chunks
Ada High watermark: 1600

1 biggest memory users at this time:
Results include bytes and chunks still allocated
Traceback elements allocated: 2480
Validity elements allocated: 1

Prg.Get_Memory_Dump [1, Prg.Allocations_Count];

Traceback elements allocated: 2798
Validity elements allocated: 1

Ada Allocs: 68456 bytes in 1419 chunks
Ada Free: 67588 bytes in 1405 chunks
Ada Current watermark: 868 in 14 chunks
Ada High watermark: 1600

1 biggest number of live allocations:
Results include bytes and chunks still allocated
5.5%: 48 bytes in 1 chunks at 0x000000000040C509 0x000000000040C33B
0x000000000043B74A 0x000000000043D42F 0x000000000042B7A7 0x0000000000407090
0x000000000040C2BE 0x0000000000474D27 0x00000000004053C8

Prg.Get_Memory_Dump [1, Prg.Sort_Total_Allocs];

Traceback elements allocated: 3106
Validity elements allocated: 1

Ada Allocs: 75816 bytes in 1573 chunks
Ada Free: 74948 bytes in 1559 chunks
Ada Current watermark: 868 in 14 chunks
Ada High watermark: 1600

1 biggest number of allocations:
Results include total bytes and chunks allocated,
even if no longer allocated - Deallocations are ignored

Prg.Get_Memory_Dump [1, Prg.Marked_Blocks];

Traceback elements allocated: 3414
Validity elements allocated: 1

Ada Allocs: 83192 bytes in 1727 chunks
Ada Free: 82324 bytes in 1713 chunks
Ada Current watermark: 868 in 14 chunks
Ada High watermark: 1600

Special blocks marked by Mark_Traceback
0.0%: 0 chunks / 1 at 0x000000000040C509 0x000000000040C33B 0x000000000043B74A
0x000000000043DB1E 0x00000000004126A5 0x000000000041AC80 0x000000000041ED3D
0x0000000000405B71 0x000000000040C2BE 0x0000000000474D27 0x00000000004053C8

9.7 Get_System_Name

• Description

Returns system name like "Debian" or "Ubuntu" or "System not handled [unprocessed system string returned]".

- Usage

```
function Get_System_Name return VString;
```

- Example

```
Get_System_Name - for system "Debian 11 GNU/Linux 11"
"debian"
```

9.8 Get_System_Version

- Description

Returns system version like 10, 11 for Debian or 18.04, 20.04, 22.04 for Ubuntu or "System not handled [unprocessed system string returned]". For Ubuntu systems, subversion like 18.04.6 and **LTS** string are omitted.

- Usage

```
function Get_System_Version return VString;
```

- Example

```
Get_System_Version - for system "Debian 11 GNU/Linux 11"
"11"

Get_System_Version - for system "Ubuntu 22.04 LTS"
"22.04"
```

9.9 Install_Packages

- Description

Install packages for Debian, Ubuntu or derivatives distributions.

- Usage

```
function Install_Packages (Packages_List : String; Host_Name : VString := "") re-
turn Boolean;
function Install_Packages (Packages_List : VString; Host_Name : VString := "") re-
turn Boolean;
```

- Example

```
if not Sys.Install_Packages ["curl, libtool, libcurl4, libcurl4-openssl-dev,
libssl-dev"] then
  Log.Err ["At least one package has not been installed. "];
end if;
```

9.10 Is_Command

- Description

Return true if command exists and reachable from path.

- Usage

```
function Is_Command [Package_Name : String] return Boolean  
function Is_Command [Package_Name : VString] return Boolean
```

- Example

```
if not Is_Command ["mc"] then  
  Log.Err ["Midnight Commander not available."];  
end if;
```

9.11 Is_Package

- Description

Return true if Package_Name is installed.

- Usage

```
function Is_Package [Package_Name : String] return Boolean
```

- Example

```
if not Is_Package ["curl"] then  
  Log.Err ["Package Curl is missing."];  
end if;
```

9.12 Purge_Packages

- Description

Purge packages for Debian, Ubuntu or derivatives distributions.

- Usage

```
function Purge_Packages [Packages_List : String; Host_Name : VString := +"" ] return  
Boolean;  
function Purge_Packages [Packages_List : VString; Host_Name : VString := +"" ] re-  
turn Boolean;
```

- Example

```
if not Sys.Purge_Packages ["exim4-base, exim4-config, exim-4-daemon-light"]
then
  Log.Err ["At least one package has not been purged."];
end if;
```

9.13 Reset_Memory_Monitor

- Description

Reset all internal data [i.e. reset all displayed counters. This is in general not needed, unless you want to know what memory is used by specific parts of your application.

➤ You must activate memory monitor with Set_Memory_Monitor before using this function.

- Usage

procedure Reset_Memory_Monitor

- Example

```
Reset_Memory_Monitor;
```

9.14 Set_Env

- Description

Set an environment variable Name.

- Usage

```
procedure Set_Env [Name : String; Value : String]
procedure Set_Env [Name : VString; Value : String]
procedure Set_Env [Name : String; Value : VString]
procedure Set_Env [Name : VString; Value : VString]
```

- Example

<<<TODO>>>

9.15 Set_Memory_Monitor

- Description

If Activate_Monitor is true, the program will monitor all memory allocations and deallocations, and through the Get_Memory_Dump procedure below be able to report the memory usage. The overhead is almost null when the monitor is disabled.

- Usage

```
procedure Set_Memory_Monitor (State : Boolean := True)
```

- Example

Activate memory monitor :

```
Prg.Set_Memory_Monitor;
```

Disable memory monitor :

```
Prg.Set_Memory_Monitor [False];
```

9.16 Shell_Execute

- Description

Executes shell command. Return the exit code if passed from the executed command. Without Output parameter, the command console output is displayed by default but can be redirected. If Output is used, then the executed command output is return in this parameter.

- Usage

```
procedure Shell_Execute (Command : String)
procedure Shell_Execute (Command : VString)
procedure Shell_Execute (Command : String; Result : out Integer)
procedure Shell_Execute (Command : VString; Result : out Integer)
procedure Shell_Execute (Command : String; Result : out Integer; Output : out
VString)
procedure Shell_Execute (Command : VString; Result : out Integer; Output : out
VString)
```

- Example

```
-----
declare
  SE_Result : Integer := 0;
begin
  Sys.Shell_Execute ["find test.cfg", SE_Result];
  Tio.Put_Line(SE_Result);
  Tio.Line;
end;

0 <- found

-----
declare
  SE_Result : Integer := 0;
begin
  Sys.Shell_Execute ["find i.dont.exist", SE_Result];
  Tio.Put_Line(SE_Result);
```

```

    Tio.Line;
end;

1 <- not found

-----
declare
    SE_Result : Integer := 0;
    SE_Output : VString := +"";
begin
    Sys.Shell_Execute ["cat test.cfg", SE_Result, SE_Output];
    if SE_Result = 0 then
        Tio.Put_Line [SE_Output];
        Tio.Line;
    end if;
end;

[Section_1]
Parameter_11 = Value_11
[Section_2]
Parameter_21 = Value_21
[Section_3]
Parameter_31 = Value_31

...which is the content of test.cfg.

```

10 Tio - Text console

```

Max_Row : constant Natural := 24;
Max_Column : constant Natural := 79;

subtype Row is Natural range 0..Max_Row;
subtype Column is Natural range 0..Max_Column;

```

10.1 Animated_Delay

- Description

Animated delay in seconds with markers each 5 and 10 seconds.

- Usage

```
procedure Animated_Delay [Delay_Seconds : Positive];
```

- Example

```

Animated_Delay [27];

....!....|....!....|../ < animated wheel with /-\|/-| characters
.1s !5s |10s

When finished

....!....|....!....|....!..

```

10.2 Ansi

- Description

Get and set ANSI state for v20 display functions and procedures.

- Usage

Tio.Ansi := False/True;

- Example

<<<TODO>>>

10.3 Beep

- Description

Send a beep.

- Usage

procedure Beep

- Example

<<<TODO>>>

10.4 Clear_Screen

- Description

Clear the screen.

- Usage

procedure Clear_Screen

- Example

```
-- Clear the screen  
Clear_Screen;
```

10.5 Confirm_Twice

- Description

Double check by user before action. Returns True if user has validate.

- Usage

function Confirm_Twice [User_Prompt_1 : VString ; User_Prompt_2 : VString] re-
turn Boolean;

- Example

```
if Tio.Confirm_Twice [Name & ".Create: Do you want to create the " & Name,  
                    Name & ".Create: Confirm creation of " & Name] then  
    ... Some operations ...  
    Tio.Put_Line [Name & ".Create: " & Name & " " & Value & " created"];  
end if;
```

10.6 Cursor_Line_Backward

- Description

Move the cursor backward X rows.

- Usage

procedure Cursor_Line_Backward [X : Row]

- Example

<<<TODO>>>

10.7 Cursor_Line_Erase

- Description

Erase the current line from the current cursor position to the end of the line.

- Usage

procedure Cursor_Line_Erase [X : Row]

- Example

<<<TODO>>>

10.8 Cursor_Line_Forward

- Description

Move the cursor forward X rows.

- Usage

procedure Cursor_Line_Forward [X : Row]

- Example

<<<TODO>>>

10.9 Cursor_Line_Move

- Description

Move the cursor at the specified X,Y coordinates.

- Usage

procedure Cursor_Move [X : Row; Y : Column]

- Example

<<<TODO>>>

10.10 Cursor_Off

- Description

Hide the cursor console.

- Usage

procedure Cursor_Off

- Example

Cursor_Off;

10.11 Cursor_On

- Description

Display the cursor console.

- Usage

procedure Cursor_On

- Example

Cursor_On;

10.12 Cursor_Restore

- Description

Restore the previous saved cursor position.

- Usage

procedure Cursor_Restore

- Example

<<<TODO>>>

10.13 Cursor_Save

- Description

Save the current cursor position.

- Usage

procedure Cursor_save

- Example

<<<TODO>>>

10.14 Line

- Description

Create a new blank line, or more than one when Spacing is passed.

- Usage

procedure New_Line [Spacing : Positive]

- Example

<<<TODO>>>

10.15 Get_Immediate

- Description

Get a character validated by [Enter]

- Usage

procedure Get_Immediate [C : out Character]

- Example

```

procedure Pause is
  Dummy : Character;
begin
  Put_Line ["Press any key to continue..."];
  Get_Immediate{Dummy};
end Pause;
```

10.16 Get_Password

- Description

Returns a password blind typed

- Usage

function Get_Password return VString

- Example

```
Pass := Get_Password;  
Password:
```

10.17 Pause

- Description

Displays *Press any key to continue or [Ctrl-C] to abort...* waiting for user input.

- Usage

procedure Pause

- Example

```
procedure Test_Pause is  
begin  
  Pause;  
end Test_Pause;
```

10.18 Put

- Description

Print to the console.

- Usage

```
procedure Put {B : Boolean};  
procedure Put {I : Integer};  
procedure Put {I : Long_Integer};  
procedure Put {C : Character}  
procedure Put {S : String};  
procedure Put {V : VString};
```

- Example

<<<TODO>>>

10.19 Put_Line

- Description

Print to the console then add a new line.

- Usage

```
procedure Put_Line [B : Boolean];  
procedure Put_Line [I : Integer];  
procedure Put_Line [L : Long_Integer];  
procedure Put_Line [C : Character];  
procedure Put_Line [S : String];  
procedure Put_Line [V : VString];
```

- Example

<<<TODO>>>

11 Tio - Text files

```
subtype File is Ada.Text_IO.File_Type;  
Copy_Form : constant String := "preserve=no_attributes,mode=overwrite";
```

11.1 Append

- Description

Append a file.
File mode is "Out" [write mode].

- Usage

```
procedure Append [Handle : in out File; Name : String]  
procedure Append [Handle : in out File; Name : VString]
```

- Example

```
Append [File_Tmp_Handle, +". /toto"];  
while not End_Of_File [File_Tmp_Handle] loop  
  Put_Line [File_Tmp_Handle, +"This is a new line of data"];  
end loop;  
Close [File_Tmp_Handle];
```

11.2 Close

- Description

Close a file.

- Usage

procedure Close [Handle : in out File]

- Example

```
Open [File_Tmp_Handle, +". /toto"];
while not End_Of_File [File_Tmp_Handle] loop
  Get_Line [File_Tmp_Handle, Line_Buffer];
end loop;
Close [File_Tmp_Handle];
```

11.3 Create

- Description

Create a file.

File mode is "Out" [write mode].

- Usage

procedure Create [Handle : in out File; Name : String]
 procedure Create [Handle : in out File; Name : VString]

- Example

```
.../...
  File_Tmp_Handle : Tio.File;
begin
  Create [File_Tmp_Handle, +". /toto"];
  Put_Line [File_Tmp_Handle, "Write a first line in ./toto"];
  Put_Line [File_Tmp_Handle, "Write a second line in ./toto"];
  Close [File_Tmp_Handle];
.../...
```

11.4 End_Of_Line

- Description

Return true if end of line is reached.

- Usage

function End_Of_Line [Handle : File] return Boolean
 function End_Of_Line [Handle : File] return Boolean

- Example

<<<TODO>>>

11.5 End_Of_File

- Description

Return true if end of file is reached.

- Usage

```
function End_Of_File (Handle : File) return Boolean
function End_Of_File (Handle : File) return Boolean
```

- Example

<<<TODO>>>

11.6 Flush

- Description

Flush file buffer to disk.

- Usage

```
procedure Flush (Handle : in File)
```

- Example

<<<TODO>>>

11.7 Get

- Description

Get the current line.

- Usage

```
procedure Get (Handle : File; C : out Character)
procedure Get (Handle : File; S : out String)
procedure Get (Handle : File; I : out Integer);
procedure Get (Handle : File; F : out Real);
```

- Example

```
Create (File_Tmp_Handle, +". /toto");
while not End_Of_File (File_Tmp_Handle) loop
  Get (File_Tmp_Handle, Line_Buffer);
```

```
Skip_Line;  
end loop;  
Close [File_Tmp_Handle];
```

11.8 Get_Line

- Description

Get the current line and then move the file pointer to the next line.

- Usage

procedure Get_Line [Handle : File; V : out VString]

- Example

```
Create [File_Tmp_Handle, +". /toto"];  
while not End_Of_File [File_Tmp_Handle] loop  
  Get_Line [File_Tmp_Handle, Line_Buffer];  
end loop;  
Close [File_Tmp_Handle];
```

11.9 Is_Open

- Description

Returns true if Handle file is open.

- Usage

function Is_Open [Handle : in File] return Boolean

- Example

<<<TODO>>>

11.10 Line

- Description

Create a new blank line, or more when Spacing is passed.

- Usage

procedure New_Line [Handle : File; Spacing : Positive]

- Example

<<<TODO>>>

11.11 Open_Conf

- Description

Special Open function for config files and others valuable files.

Ensure that the complete directory tree structure exists before creating file. Creating this directory tree if needed. Creates or Append files if needed.

Always make backup before Append. If Wipe_Before_Process is True, the file Name is backup-ed before being deleted.

- Usage

```
procedure Open_Conf [Handle : in out File; Name : String ;  
                    Wipe_Before_Process : Boolean := False ;  
                    Permissions : VString := +""];  
procedure Open_Conf [Handle : in out File; Name : VString ;  
                    Wipe_Before_Process : Boolean := False ;  
                    Permissions : VString := +""];
```

- Example

```
.../...  
  
    File_Tmp_Handle : Tio.File;  
  
begin  
  
    Open_Conf [File_Tmp_Handle, +". /toto", True, +"0600"];  
  
    Put_Line [File_Tmp_Handle, "Write a first line in ./toto"];  
    Put_Line [File_Tmp_Handle, "Write a second line in ./toto"];  
  
    Close [File_Tmp_Handle];  
  
    .../...
```

11.12 Open_Read

- Description

Open a file. File mode is "In" [read mode].

- Usage

```
procedure Open_Read [Handle : in out File; Name : String]  
procedure Open_Read [Handle : in out File; Name : VString]
```

- Example

```
.../...  
  
    File_Tmp_Handle : Tio.File;
```

```

begin
  Open_Read [File_Tmp_Handle, +". /toto"];
  while not End_Of_File [File_Tmp_Handle] loop
    Get_Line [File_Tmp_Handle, Line_Buffer];
  end loop;
  Close [File_Tmp_Handle];
  .../...

```

11.13 Put

- Description

Write to a file

- Usage

```

procedure Put [Handle : File; C : Character]
procedure Put [Handle : File; S : String]
procedure Put [Handle : File; V : VString]

```

- Example

<<<TODO>>>

11.14 Put_Line

- Description

Write a file and then add a new line

- Usage

```

procedure Put_Line [Handle : File; C : Character]
procedure Put_Line [Handle : File; S : String]
procedure Put_Line [Handle : File; V : VString]

```

- Example

<<<TODO>>>

11.15 Read_File

- Description

Read a text file File_To_Read and returning a VString buffer. LF [line feed] are preserved.

- Usage

```

function Read_File [File_Name : VString] return VString

```

- Example

<<<TODO>>>

11.16 Reset

- Description

Reset the file pointer to the start of the file

- Usage

procedure Reset [Handle : in out File]

- Example

<<<TODO>>>

11.17 Write_File

- Description

Write a text file File_To_Write with Content. LF in content are preserved and used as line feed. Read Open_Conf documentation for implementation details.

- Usage

procedure Write_File [File_Name : VString, Content : VString ; Permissions : VString := +""]

- Example

<<<TODO>>>

12 Vst - VStrings

Variable-size string type

Null_VString : VString

12.1 Char_Count

- Description

Count each char in String_To_Process relative to Char_Set_Pattern.

- Usage

function Char_Count [String_To_Process : VString ; Char_Set_Pattern : String] re-
turn Integer;
function Char_Count [String_To_Process : VString ; Char_Set_Pattern : VString] re-
turn Integer;

- Example

```
Tio.Put_Line [+"alpha", "ap" ];
3
```

12.2 Element

- Description

Return the Character in Index position of the VString argument.
Index starts at one.

- Usage

function Element [Source : VString; Index : Positive] return Character

- Example

<<<TODO>>>

12.3 Empty

- Description

Return True if String or VString Source is empty.

- Usage

function Empty [Source : String] return Boolean;
function Empty [Source : VString] return Boolean;

- Example

```
Tio.Put_Line [Empty [+""] ];
True
```

12.4 Ends_With

- Description

Check if VString Item ends with another VString or String Pattern.

- Usage

function Ends_With [Item : VString; Pattern : Character] return Boolean;
function Ends_With [Item : VString; Pattern : String] return Boolean
function Ends_With [Item : VString; Pattern : VString] return Boolean

- Example

```
- Check VString with String pattern
if Ends_With ["package", "age"] then
  Put_Line ["Match !"];
end if;

- Check VString with VString pattern
if Ends_With ["package", "age"] then
  Put_Line ["Match !"];
end if;
```

12.5 Field_* functions guidelines

Field_* functions deal with string [String_To_Process] forming lists of fields separated by a delimiting character [Field_Delimiter].

Use only Field_Delimiter characters between 0dec and 127dec, due to some keyboard available characters encoding with 2 chars.

Some recommended Field_Delimiter characters are listed in v20.ads but also above in the v20 documentation : Delimiter characters.

12.6 Field_By_Index

- Description

Return a field indexed by Index_Field and delimited by Field_Delimiter.

- Usage

```
function Field_By_Index (String_Input : VString ; Index_Field : Integer ; Field_Delimiter : String) return VString;
```

- Example

```
Tio.Put_Line [Field_By_Index ["alpha:bravo:charlie", 2, ":"]];

bravo
```

12.7 Field_By_Name

- Description

Return a field from a search string and delimited by Field_Delimiter. Returns an empty VString if not found.

- Usage

```
function Field_By_Name (String_Input : VString ; Field_To_Search : VString ; Field_Delimiter : String) return VString;
```

- Example

```
Tio.Put_Line [Field_By_Name ["alpha:bravo:charlie", "rav", ":"]];
bravo
```

12.8 Field_Count

- Description

Count fields in String_To_Process and return fields number.

➤ To handle one field case without trailing Field_Delimiter, if String_To_Process not empty and Field_Delimiter not found, Field_Count returns 1.

- Usage

function Field_Count (String_To_Process : VString ; Field_Delimiter : String) return Integer;

- Example

```
Tio.Put_Line [Field_Count ["alpha:bravo:charlie", ":"]];
3
```

12.9 Field_Included

- Description

Returns True if all Items_List are included in String_To_Process list, which is delimited by Field_Delimiter.

- Usage

function Field_Included (String_To_Process : VString ; Items_List : Vstring ; Field_Delimiter : String) return Boolean;

- Example

```
Tio.Put_Line [Field_Count ["alpha,bravo,charlie", "alpha,charlie" , ",", "]];
True
```

12.10 Field_Display

- Description

Formatted display of a string fields structured in rows and columns. Optional header names are separated by commas.

Constants declaration abstract in v20 [related to Field_* functions] :

```
ND : constant String := "~"; -- Name/value delimiter
CD : constant String := "^"; -- Column delimiter
RD : constant String := "\"; -- Row delimiter
```

- Usage

```
procedure Field_Display [String_To_Process : VString; Column_Delimiter : String;
Row_Delimiter : String; Custom_Header : String := ""];
```

- Example

Combined example with Vst.Field_Display and Sql.Read functions :

```
Field_Display [Sql.Read ["Cluster", "Number, Domain"], CD, RD, "Cluster num-
ber, Domain name"];
```

```
Cluster number  Domain name
-----
1              domain1
2              domain2
3              domain3
4              domain4
1234          genesix2.org
```

12.11 Field_Search

- Description

Search Field_To_Search in String_To_Process and return True if found.

- Usage

```
function Field_Search [String_To_Process : VString ; Field_To_Search : VString ;
Field_Delimiter : VString] return Boolean;
```

- Example

```
Tio.Put_Line [Field_Search ["alpha:bravo:charlie", "bravo", ":"]];
True
```

12.12 Head

- Description

Extract a VString between the beginning to Count Value to a VString.
Count starts at one.

- Usage

```
function Head [Source : VString; Count : Natural] return VString
```

- Example

```
Put_Line [Head ["ABCDEFGH", 4]];

"ABCD"
```

12.13 Index

- Description

Returns Natural start position of String or VString Pattern in the target Vstring
Source, From a starting index.
Natural is zero if not found.
Natural starts at one.

- Usage

```
function Index [Source : VString; Pattern : Character] return Natural;
function Index [Source : VString; Pattern : String] return Natural
function Index [Source : VString; Pattern : VString] return Natural
function Index [Source : VString; Pattern : String; From : Natural] return Natural
function Index [Source : VString; Pattern : VString; From : Natural] return Natural
```

- Example

```
if Index ["ABCDABCD", "BC"] = 2 then
  Put_Line ["Match !"];
end if;

if Index ["ABCDEFGH", "BC", 4] = 6 then
  Put_Line ["Match !"];
end if;
```

12.14 Index_backward

- Description

From the end of the target Vstring Source, returns Natural start position of String
or VString Pattern in the target Vstring Source, From a backward starting index.
Natural is zero if not found.
Natural starts at one.

- Usage

```
function Index_Backward [Source : String; Pattern : String] return Natural;  
function Index_Backward [Source : VString; Pattern : String] return Natural  
function Index_Backward [Source : VString; Pattern : VString] return Natural  
function Index_Backward [Source : VString; Pattern : String; From : Natural] return  
Natural  
function Index_Backward [Source : VString; Pattern : VString; From : Natural] re-  
turn Natural
```

- Example

```
if Index_Backward ["abcdefabcdef", "cd"] = 9 then  
  Put_Line ["Match !"];  
end if;  
  
if Index_Backward ["abcdefabcdef", "cd", 8] = 3 then  
  Put_Line ["Match !"];  
end if;
```

12.15 Length

- Description

Returns the length of the String or VString represented by Source.

- Usage

```
function Length [Source : String] return Natural;  
function Length [Source : VString] return Natural
```

- Example

```
Put [Length ["ABCDEFGH"]];  
  
8
```

12.16 Replace_Char

- Description

Replace all Char_In by Char_Out in String_To_Process.

- Usage

```
function Replace_Char [String_To_Process : VString ; Char_In : Character ; Char_Out  
: Character] return VString;
```

- Example

```
Replace_Char ["ABCDEFGH", 'D', 'Z'];
```

"ABCZEF GH"

12.17 Replace_Pattern

- Description

Replace Pattern_In by Pattern_Out in String_To_Process. Returns a VString with Pattern_In replaced by Pattern_Out.

- Usage

```
function Replace_Pattern [String_To_Process : VString ; Pattern_In : VString ; Pattern_Out : VString] return VString;
```

- Example

```
Replace_Pattern ["ABCDEF GH", "BCD", "xyyz"];
"AxyyzDEF GH"
```

12.18 Slice

- Description

Returns a Vstring portion of the Vstring represented by Source delimited by From and To. From and To index start at one. Omitting High stands for High equal to length of source

- Usage

```
function Slice [Source : VString; From : Positive; To : Natural] return VString
```

- Example

```
Put_Line [Slice ["ABCDEF GH", 2, 4]];
"BCDE"
Put_Line [Slice ["ABCDEF GH", 2]];
"BCDEF GH"
```

12.19 Starts_With

- Description

Check if VString Item starts with another VString or String Pattern.

- Usage

```
function Starts_With [Item : VString; Pattern : Character] return Boolean;
function Starts_With [Item : VString; Pattern : String] return Boolean
```

function Starts_With [Item : VString; Pattern : VString] return Boolean

- Example

```
- Check VString with String pattern
if Ends_With ["package", "pac"] then
  Put_Line ["Match !"];
end if;

- Check VString with VString pattern
if Ends_With ["package", "pac"] then
  Put_Line ["Match !"];
end if;
```

12.20 Stript_Accents

- Description

Replace common accented characters with their lower ASCII equivalent. Encoding processed are Latin_1, UTF-8 and character handled are à â é è ê ë î ï ô ù ç.

- Usage

function Stript_Accents [String_To_Process : VString] return VString

- Example

```
Put_Line [Sript_Accents ["île d'oléron"]];
"île d'oleron"
```

12.21 Stript_Chars

- Description

Sript each char in String_To_Process relative to Char_List.

- Usage

function Stript_Chars [String_To_Process : VString ; Char_List : VString] return VString

- Example

```
Put_Line [Sript_Chars ["ABCDEFGH", "BDF"]];
"ACEGH"
```

12.22 Tail

- Description

Extract a VString from Source between its end to backward Count Value. Count starts at one [backward].

- Usage

function Tail [Source : VString; Count : Natural] return VString

- Example

```
Put_Line [Tail ["ABCDEFGH", 4]];
"EFGH"
```

12.23 Tail_After_Match

- Description

Extract a VString from Source starting from Pattern+1 position to the end.

- Usage

```
function Tail_After_Match [Source : VString; Pattern : Character] return VString;
function Tail_After_Match [Source : String; Pattern : String] return VString;
function Tail_After_Match [Source : VString; Pattern : String] return VString;
function Tail_After_Match [Source : VString; Pattern : VString] return VString;
```

- Examples

```
Path := "/etc/genesix/gnx-startup";

Put_Line [Tail_After_Match [Path, '/' ]];
"gnx-startup"

Put_Line [Tail_After_Match [Path, "ix"]];
"/gnx-startup"

Put_Line [Tail_After_Match [Path, "gene"]];
"six/gnx-startup"

Put_Line [Tail_After_Match [Path, "etc/genesix/gnx-startu"]];
"p"

Put_Line [Tail_After_Match [Path, "/etc/genesix/gnx-startu"]];
"p"

Put_Line [Tail_After_Match [Path, "/etc/genesix/gnx-startup"]];
empty string

Put_Line [Tail_After_Match [Path, +"/etc/genesix/gnx-startupp"]];
empty string

Put_Line [Tail_After_Match [Path, +"/etc/geneseven"]];
empty string
```

12.24 To_Lower

- Description

Convert a Character or a VString to lower case.

- Usage

```
function To_Lower [Item : Character] return Character
function To_Lower [Item : String] return VString
function To_Lower [Item : VString] return VString
```

- Example

<<<TODO>>>

12.25 To_Upper

- Description

Convert a Character or a VString to upper case.

- Usage

```
function To_Upper [Item : Character] return Character
function To_Upper [Item : String] return VString
function To_Upper [Item : VString] return VString
```

- Example

<<<TODO>>>

12.26 Trim_Both

- Description

Returns an all trimmed spaces VString of VString Source.

- Usage

function Trim_Both [Source : VString] return VString

- Example

```
Put_Line [Trim_Right [ "+" AB CD " ]];
"AB CD"
```

12.27 Trim_Left

- Description

Returns a trimmed leading spaces VString of VString Source.

- Usage

function Trim_Left [Source : VString] return VString

- Example

```
Put_Line [Trim_Left [ "+" ABCD " ]];
"ABCD "
```

12.28 Trim_Right

- Description

Returns a trimmed trailing spaces VString of VString Source.

- Usage

function Trim_Right [Source : VString] return VString

- Example

```
Put_Line [Trim_Right [ "+" ABCD " ]];
" ABCD"
```

12.29 Trim_Slashes

- Description

Returns an all trimmed slashes VString of VString Source.

- Usage

function Trim_Slashes [Source : VString] return VString

- Example

```
Trim_Slashes { "/" }
""

Trim_Slashes { "I" }
"I"

Trim_Slashes { "/i" }
"I"

Trim_Slashes { "////////i////////" }
"i"
```

12.30 +

- Description

Cast a String to a VString.

- Usage

function "+" [C : Character] return VString renames To_VString;
function "+" [S : String] return VString

- Example

<<<TODO>>>

12.31 *

- Description

Duplicate a Character, String or VString Num times to a VString.

- Usage

function "*" [Num : Natural; Pattern : Character] return VString
function "*" [Num : Natural; Pattern : String] return VString
function "*" [Num : Natural; Pattern : VString] return VString

- Example

```
Put_Line {3 * "0"};
```

```
"000"
Put_Line [3 * +"12"];
"121212"
```

12.32 &

- Description

Concatenate a VString with a VString, String, Character, Integer and Real to a VString

- Usage

```
function "&" [V1, V2 : VString] return VString
```

```
function "&" [V : VString; S : String] return VString
function "&" [S : String; V : VString] return VString
```

```
function "&" [V : VString; C : Character] return VString
function "&" [C : Character; V : VString] return VString
```

```
function "&" [I : Integer; V : VString] return VString
function "&" [V : VString; I : Integer] return VString
```

```
function "&" [R : Real; V : VString] return VString
function "&" [V : VString; R : Real] return VString
```

12.33 =

- Description

Test equality between a VString and another VString or String.

- Usage

```
function "=" [Left, Right : VString] return Boolean
function "=" [Left : VString; Right : String] return Boolean
function "=" [Left : String; Right : VString] return Boolean
```

- Example

<<<TODO>>>

12.34 <

- Description

<<<TODO>>>

- Usage

```
function "<" [Left, Right : VString] return Boolean
function "<" [Left : VString; Right : String] return Boolean
function "<" [Left : String; Right : VString] return Boolean
```

- Example

<<<TODO>>>

12.35 <=

- Description

<<<TODO>>>

- Usage

```
function "<=" [Left, Right : VString] return Boolean
function "<=" [Left : VString; Right : String] return Boolean
function "<=" [Left : String; Right : VString] return Boolean
```

- Example

<<<TODO>>>

12.36 >

- Description

<<<TODO>>>

- Usage

```
function ">" [Left, Right : VString] return Boolean
function ">" [Left : VString; Right : String] return Boolean
function ">" [Left : String; Right : VString] return Boolean
```

- Example

<<<TODO>>>

12.37 >=

- Description

- Usage

```
function ">=" [Left, Right : VString] return Boolean
function ">=" [Left : VString; Right : String] return Boolean
function ">=" [Left : String; Right : VString] return Boolean
```

- Example

<<<TODO>>>

13 Vst - Types conversion and tests

13.1 Ascii_Value_To_Hex

- Description

Convert an ASCII VString value ranging 0..127 to a VString hexadecimal output.

- Usage

function Ascii_Value_To_Hex [Input : VString] return VString

- Example

```
tio.Put_Line [vst.Ascii_Value_To_Hex ["61"]];
3D
```

13.2 Is_Numeric

- Description

Return True if Item string is numeric.

- Usage

function Is_Numeric [Item : in String] return Boolean;
function Is_Numeric [Item : in VString] return Boolean;

- Example

```
tio.Put_Line [vst.Is_Numeric ["12AZE12"]];
False
tio.Put_Line [vst.Is_Numeric ["1212"]];
True
```

13.3 To_Hex

- Description

Convert a Byte or VString to a String or VString hexadecimal output.

- Usage

function To_Hex [B : Byte] return String

```
function To_Hex [V : VString] return VString
```

- Example

```
tio.Put_Line [vst.To_Hex ["ABCDEF"]];  
41 42 43 44 45 46
```

13.4 To_Integer

- Description

Convert a String or VString to an Integer.
Leading and trailing spaces are trimmed before conversion.
Returns 0 if String or VString is empty or contains non numeric character.

- Usage

```
function To_Integer [V : String] return Integer  
function To_Integer [V : VString] return Integer
```

- Example

<<<TODO>>>

13.5 To_String

- Description

Convert a VString to a String.

- Usage

```
function To_String [V : VString] return String
```

- Example

<<<TODO>>>

13.6 To_Val

- Description

Convert a VString to VString ASCII decimal formatted output.

- Usage

```
function To_Val [V : VString] return VString
```

- Example

```
tio.Put_Line [vst.To_Val ["ABCDEF"]];
```

13.7 To_VString

- Description

Convert a Boolean, an Integer, a Char or a String type into VString type.

- Usage

```
function To_VString [B : Boolean] return VString  
function To_VString [I : Integer] return VString  
function To_VString [C : Character] return VString  
function To_VString [S : String] return VString
```

- Example

```
Input : String := "ABC";  
Result : VString;  
Result := To_VString [Input];
```

v20 architecture

Doubling the number of programmers on a late project does not make anything else than double the delay.

Second Brook's Law



1 Introduction

<<<TODO>>>

2 Requirements

An Ada compiler from the GCC/GNAT family, preferably a GNAT CE 2020.

An Unix system, preferably a GNU/Linux Debian [or Debian based like Ubuntu or Mint].

3 Coding guidelines

3.1 General

Language: English

Source code length: 79 columns

Naming: Capitalize and use underscore with compound name. ex: Entry_Value

3.2 Messages

□ Log.Msg ["Blahblah."]

Information messages start with a capital and end with a dot. Ending messages with three dots are only allowed when a user input is waited.

□ Log.Err ["v20.Fls.Function_Name - Can't do something."]

Error messages start with the library or program hierarchy followed by a dash and then the error message.

3.3 Naming

We tried to avoid few naming or consistency flaws of the original Ada runtime:

- The text mode *Open* function of v20 now logically opens in *File_In* mode (read mode);
- If the procedures *Put* and *Put_Line* are named like this, then *New_Line* should be called *Line* :)

4 Design

v20 is designed as a KISS working library. It does not attempt to reproduce the outstanding granularity of the Ada runtime.

<<<TODO>>>

4.1 Types

Name	Packages	Description
Character	Base	
String	VString	
VString	Program	Unbounded string subtyping from HAC by Gautier de Montmollin
Integer	Text I/O	Text Input/Output related
Boolean	Logging	Log - Terminal and file log - on top of Tio
BCD		Financial computing
Float		Scientific computing
Geo		Geo. Coords.
	handling ok	

4.2 Packages

Name	Packages	Description
v20	Base	
Bio	Binary I/O	Binary IO: Binary files, locking, etc.
Cfg	Configuration files	Simple and user friendly config files handling
Dbf	Multiusers btree DB	Data base files: indexed btree with locks management - on top of Bio
Eml	Email	Pop3/Smtp
Fls	File system	
Log	Logging	Log - Terminal and file log - on top of Tio
Net	Network	
Pdf	Pdf handling	See Gautier de Montmollin package
Prg	Program	Program and user related
Prt	Printer package	Print to local network duplex A3 & A4 printer [see previous works: v90, psrc and a2ps]
Rts	Run Time System	AVR embedded
Ser	Serial handling	Tx, Rx and spying
Sql	SQL database	SQLite high level implementation
Sys	System	Operating System related
Tio	Text I/O	Text Input/Output related

Name	Packages	Description
Usb	Usb handling	Tx, Rx and spying
Vst	VString	Unbounded string subtyping from HAC by Gautier de Montmollin
	Already coded	

4.3 Functions

About strings, v20 functions always [*should* actually] return VString [never String type].

<https://this-page-intentionally-left-blank.org>



FAQ

With the Wildebeest and the Penguin, there's no Bull.
Number Six



1 Conventional exit codes

- 1 -h or --help switches
- 2 invalid switch
- 3 invalid parameter
- 4 SQL error
- 5 reserved for future use
- 6 reserved for future use
- 7 reserved for future use
- 8 reserved for future use
- 9 if an exception occurs during execution

Exit codes greater than 9 are reserved to applications using v20. Typically, an application may use a base exit code by class command with local increment. Example: exit code for command "service backup" [backup being the first command of class service] could be `Base_Exit_Code_Service + 1` :

<code>Base_Exit_Code_App :</code>	<code>constant Positive := 10;</code>
<code>Base_Exit_Code_Cluster :</code>	<code>constant Positive := 20;</code>
<code>Base_Exit_Code_Db :</code>	<code>constant Positive := 30;</code>
<code>Base_Exit_Code_Domain:</code>	<code>constant Positive := 40;</code>
<code>Base_Exit_Code_Group :</code>	<code>constant Positive := 50;</code>
<code>Base_Exit_Code_Help :</code>	<code>constant Positive := 60;</code>
<code>Base_Exit_Code_Instance :</code>	<code>constant Positive := 70;</code>
<code>Base_Exit_Code_Info :</code>	<code>constant Positive := 80;</code>
<code>Base_Exit_Code_Ip :</code>	<code>constant Positive := 90;</code>
<code>Base_Exit_Code_Node :</code>	<code>constant Positive := 100;</code>
<code>Base_Exit_Code_Owner :</code>	<code>constant Positive := 110;</code>
<code>Base_Exit_Code_Remote :</code>	<code>constant Positive := 120;</code>
<code>Base_Exit_Code_Service :</code>	<code>constant Positive := 130;</code>
<code>Base_Exit_Code_User :</code>	<code>constant Positive := 140;</code>

2 Converting reminder

2.1 Converting Integer to String with Character'Val and Integer'Image

65 is ASCII code for 'A' :

```
Tio.Put_Line [Integer' Image [65]];
The string "65"

Tio.Put_Line [Character' Val[65]];
The string "A"
```

2.2 Converting a character to its ASCII value

65 is ASCII code for 'A' :

```
Tio.Put_Line [Character' Pos[' A' ]];
The string "65"
```

2.3 Converting VString from and to Long_Integer

```
procedure Test is

    Test_VString : VString := +"";
    Test_String : String := "10737418240";
    Test_Long_Integer : Long_Integer := 10737418240;

begin

    Test_VString := To_VString [Long_Integer' Image [Test_Long_Integer]];

    Log.Msg [Test_String];

    Log.Msg [Test_Long_Integer];

    -- Non qualified expression conversion
    Test_Long_Integer := Long_Integer' Value ["10737418240"];

    -- Qualified expression conversion
    Test_Long_Integer := Long_Integer' Value [String ' ["10737418240"]];

end Test;
```

3 How to prepare SQLite to v20 integration

3.1 Simple Components

```
wget http://www.dmitry-kazakov.de/ada/components_4_58.tgz
mkdir scdk ; cd scdk
http://www.dmitry-kazakov.de/ada/components_4_58.tgz
tar xzf components_4_58.tgz
```

Put files object.ad*, object-handle.ad*, sqlite.ad* in project source path.

3.2 SQLite

To avoid linker warnings when building statically, you must disable load extension to prevent the dynamic load extension.

Create sqlite3 amalgamation :

```
git clone https://github.com/sqlite/sqlite
cd sqlite
./configure --enable-static --disable-load-extension
make
```

At the very beginning of sqlite3.c, add the line :

```
#define SQLITE_OMIT_LOAD_EXTENSION 1
```

Put files sqlite3.h, sqlite3.c in project source path.

<https://this-page-intentionally-left-blank.org>



Programs examples

Weinberg's Second Law : If builders built buildings the way programmers wrote programs, then the first woodpecker that came along would destroy civilization.

Gerald Weinberg



1 test.adb

<<<TODO>>>

<https://this-page-intentionally-left-blank.org>

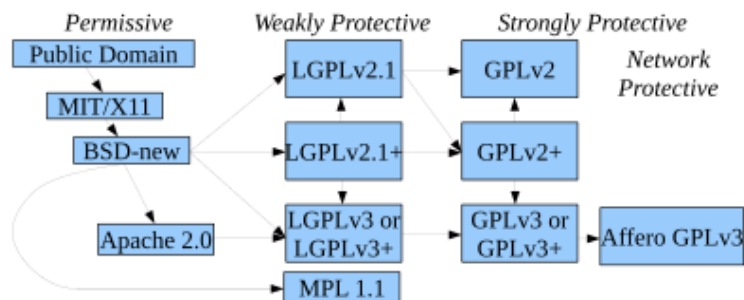


Appendices

1 Copyrights & credits

1.1 Library Licence

v20 is copyright Sowebio under GPL v3 license.



- GPL v3 compatibility with others licenses

https://en.wikipedia.org/wiki/License_compatibility: MIT licence is compatible with GPL and can be re-licensed as GPL. European Union Public Licence [EURL] is *explicitly compatible* with GPL v2 v3, OSL v2.1 v 3, CPL v1, EPL v1, CeCILL v2 v2.1, MPL v2, LGPL v2.1 v3, LiLIQ R R+ AGPL v3.

1.2 Manual license

This manual is intended for v20, a KISS library for Ada command line programs. Copyright ©2004, 2005, 2020, 2021 Stéphane Rivière. This document may be copied, in whole or in part, in any form or by any means, as is or with alterations, provided that alterations are clearly marked as alterations and this copyright notice is included unmodified in any copy.

1.3 v20 Packages copyrights & credits

Vst - Variable Strings from HAC runtime - gdm sr : HAC is copyright Gautier de Montmollin.

2 To-do list

2.1 v20.Sql

Git like sync between sqlite db.

Design notes :

```
sqlite3 database1.db
> ATTACH DATABASE 'database2.db' AS database2;
> INSERT INTO mytable SELECT * FROM database2.mytable;
```

Check si deux dbsqlite sont identiques

<https://www.sqlite.org/dbhash.html>

SQLite diff

<https://www.sqlite.org/sqldiff.html>

Comparison table par table

<https://stackoverflow.com/questions/2129717/how-to-verify-if-two-tables-have-exactly-the-same-data>

<https://stackoverflow.com/questions/50539501/in-sqlite-is-there-a-way-to-test-if-two-tables-are-identical>

<https://dwgeek.com/quick-best-way-compare-two-tables-sql.html>

2.2 v20.Tio

Add procedures Tio.Cursor_On and Cursor_Off using “tput civis” cursor invisible and “tput cnorm” cursor visible] or To hide the cursor: ESC + “?25l” and to To re-enable the cursor: ESC + “?25h” see <https://gist.github.com/fnky/458719343aab-d01cfb17a3a4f7296797>

Add functions “tput lines” and “tput cols” to get current console lines and columns values or the oneliner `echo -e "lines\ncols"|tput -S` or use <https://stackoverflow.com/questions/27902721/ioctl-tiocgwinsz-in-gnat-ada-returns-errno-25-but-c-program-work-fine> [should be better] and https://www.pegasoft.ca/resources/boblap/99_e.html

Tput overview : <https://stackoverflow.com/questions/5947742/how-to-change-the-output-color-of-echo-in-linux/20983251#20983251>

Add ANSI full color control including this work <https://github.com/mosteo/ansi-ada> https://en.wikipedia.org/wiki/ANSI_escape_code#CSI_sequences

Add function [enter] or [quit]

Add function [Yes] or [no] with Yes/No default choice

2.3 Doc

❑ The never-ending task

Hunt <<<TODO>>> tags :)

3 Quality control

Check list

<<< TODO>>>

4 Release check list

Things to do to release to github

<<< TODO >>>

5 Issues

5.1 Compiler bug reporting

Historic and still working report email: report@gnat.com

Since the beginning of the XXIth century: report@adacore.com

❑ Exception with Delete_Tree dealing with broken symbolic links

In french only: Ada.Directories.Del_Tree explose en présence d'un lien symbolique cassé dans un répertoire de l'arborescence à effacer: raised ADA.IO_EXCEPTION-S.USE_ERROR: directory tree rooted at "/home/sr/opt/gnat-2019/lib/xmlada/xmlada_input.relocatable" could not be deleted

• Demo

L'empilement général est

```
Ada.Directories.Delete_Tree > Is_Valid_Path_Name > Is_Directory Ada >
is_Directory C > adaint.c > __gnat_is_directory >
__gnat_reset_attributes > __gnat_is_directory_attr >
*__gnat_stat_to_attr* > __gnat_stat > GNAT_STAT
```

Du coté de More_Entries > Fetch_Next_Entry > readdir_gnat > Match

On arrive à un /lien symbolique cassé/ libxmlada_input_sources.so qui est /déclaré ne pas exister/ par File_Exists_Attr [C_Full_Name' Address, Attr' Access]; en 776 qui est en fait __gnat_file_exists_attr en 1668 de adaint.c qui fait référence à une structure dans adaint.h:

```
-----
struct file_attributes {
    int          error;

    /* Errno value returned by stat[]/fstat[]. If non-zero, other fields
    should be considered as invalid. */

    unsigned char exists;

    unsigned char writable;
    unsigned char readable;
    unsigned char executable;

    unsigned char symbolic_link;
    unsigned char regular;
    unsigned char directory;
}
-----
```

Qui appelle *__gnat_stat_to_attr*

Qui teste un file descripteur à -1, lien symbolique cassé je suppose...

Puis __gnat_stat qui renvoie 2 à __gnat_stat_to_attr

Avec le test suivant en 1124 de adaint.c

```
if [error == 0 || error == ENOENT]
    attr->error = 0;
```

Et dans s-oscons.ads ENOENT: constant := 2; -- File not found !

<shadok> Donc si on trouve pas le fichier, c'est qu'il n'y a pas d'erreur. </shadok>

La suite devient alors compréhensible... Le lien symbolique cassé libxmlada_input_sources.so est déclaré ne pas exister, la routine sort du répertoire courant [qu'elle croit donc vidé] pour l'effacer et explose alors quand elle tente d'effacer ce répertoire vide mais qui ne l'est pas...

- Solving

On pourrait re-coder cette fonction récursive plus simplement. Cru voir en traçant que la fonction C d'effacement récursif existe déjà... Toutefois, le mieux serait de corriger l'anomalie qui est probablement dans _gnat_stat, afin que cette fonction retourne la bonne valeur et ne confonde pas 'n'existe pas' [le fichier sur lequel pointe le lien symbolique] avec 'n'existe pas' [le fichier symbolique].



Ada, « it's stronger than you ».
Tribute to Daniel Feneuille, a legendary french Ada teacher [and much more]⁴

⁴ <http://d.feneuille.free.fr>