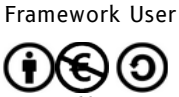
v22 Ada Framework User Manual

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

Ed.	*Release*	*Comments*	
1	20230731	Initial release	sr
18	20230821	General update, v22 & UXString Api integration	sr
22	20230824	General update, Gnoga integration	sr
34	20230912	First alpha release on private github repository	sr
36			



- Authors

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

- Acknowledgments

  v22 is based on a number of Ada open-source libraries. See Appendices for copyrights and credits.

- Manual

  Stéphane Rivière [Number Six] - stef@genesix.org [CTO Sowebio - FM1US/F1USA[1]]

  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 eaters[2]. 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 36 - 2023-09-12

[1] International amateur radio call sign - https://en.wikipedia.org/wiki/Amateur_radio.

[2] We 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 cliché* :]

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

Contents

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

Introduction

Keep It Simple, Stupid.
Clarence Leonard "Kelly" Johnson

1 About v22 framework

1.1 Ready to use in production

v22 is a general purpose, KISS[3] oriented, modular Ada framework for GNU/Linux Debian/Ubuntu[4] service, console and web programs.

v22 is composed of many packages in charge of UTF-8 strings, program and OS functions, HTTP[s]/WS[s] web framework, integrated cURL, console handling and text files, advanced network, MySQL and SQLite high level binding, logging and configuration files handling.

Originally based on the v20 library, the v22 framework represents a major step forward in the following areas:
- UTF-8 compatibility;
- Simplified string processing [only one UTF-8 String type is used];
- Internationalization;
- New and extended database API;
- Extended database access to MySQL, in addition to SQLite;
- Improved concurrent access and performance for SQLite;
- New LGPLv3 licensing instead of GPLv3;
- New FSF GCC development environment not tied anymore to GPLv2 licence;
- And much more.

1.2 Cooperative and open

v22's native dependencies are Gnoga, Simple_Components, UXStrings and Zanyblue.

v22 is both a high-level framework and an extension to the lower level components cited above. v22 has been designed to:
- Use unmodified components;

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

[4] Debian/Ubuntu features are limited to package management. Package management could be extended to other distributions. More generally, it would be possible and even desirable to adapt v22 to BSD or Mac OS systems. Any help on these subjects is welcome, as the main author of v22 only uses GNU/Linux Debian and Ubuntu.

- Not "reinvent the wheel". Component functions are to be used first;
- Offer higher-level functions or functions that do not exist in the components.

v22 must be used with the APIs of:
- UXStrings to process UTF-8 strings;
- Gnoga to connect to MySQL and/or SQLite databases;
- Gnoga to provide a Web interface;
- Simple_Components for network layer and more;
- Zanyblue if internationalization is desired.

In short:
- UXStrings is used throughout v22. The v22.Uxs package extends UXStrings functionality. The v22.Sql package extends the functionality of Gnoga.Server.Database. The v22.Gui graphics framework is based on Gnoga.Gui.
- v22's architecture allows it to be open to additional packages, depending on the software development required.

2 About the Ada Community

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

2.1 Inspiration, packages used, ideas, help and more

AdaCore Ada compiler: https://www.adacore.com/community
David Botton: https://www.linkedin.com/in/david-botton-3741b210 [Gnoga]
Dmitry A. Kazakov: http://www.dmitry-kazakov.de/ada/components.htm [Simple Components, used in Gnoga and v22]
Gautier de Montmollin: https://github.com/zertovitch [Gnoga maintainer and much more]
Jean-Pierre Rosen: https://adalog.fr [Ada teacher, writer and much more]
Jeffrey R. Carter: https://github.com/jrcarter/PragmARC [PragmAda Reusable Components]
Michael Rohan: https://sourceforge.net/projects/zanyblue [Internationalization, used in Gnoga and v22]
Pascal Pignard: https://github.com/Blady-Com [UXStrings used in Gnoga and v22, Gnoga maintainer and much more]

2.2 Special thanks

Special thanks to Ada gurus Daniel Feneuille, Gautier de Montmollin, Pascal Pignard and Jean-Pierre Rosen. The chapter heading quotes are extracted from Murphy's Law and other reasons why things go wrong - A. Bloch. They come from https://www.adalog.fr site created by Jean-Pierre Rosen.

3 v22 history

We own the copyrights for v89, v90, v93, v95, v04, v20 and v22.

Some work in v22 framework is derived from theses libraries.

Ver.	Languages	Processor	OS	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	GMGPL	Education
v20	Ada	x86-64	Linux	Sowebio	GPL v3	Terminal GP
v22	Ada	x86-64	Linux	Sowebio	LGPL v3	Terminal & Web GP

Setup

Doubling the number of programmers on a late project does not make anything else than double the delay.
Second Brook's Law

1 Dependencies

1.1 Ada dependencies

v22 mainly depends on Gnatcoll, Gnoga, UXStrings and Zanyblue.

Gnoga mainly depends on Simple_Components, UXStrings and Zanyblue.

So there's a strong community of dependencies between v22 and Gnoga.

1.2 System Dependencies

Most of the packages below are only required for Gnoga tests and demonstrations.

v22 uses its own source version of SQLite and doesn't need the libsqlite2-* packages.

MySQL support in v22 requires libmariadb3, libmariadb-dev and libmariadb-dev-compat for connection to a MariaDB database, which is preferred to MySQL, whose support on Debian and Ubuntu systems is considered rather low. But the final choice is yours.

Although this documentation always refers to MySQL, v22 is compatible with both MySQL, through its native libmysqlclient, and MariaDB, thanks to the libmariadb-dev-compat package, which emulates the libmysqlclient layer.

v22 needs libcurl4 and libcurl4-openssl-dev if v22.Crl package is used.

Finally, the libgnutls30 package is required to implement a TLS connection such as the https and wss protocols. If your application is behind a TLS proxy, this library is no longer needed.

The v22 framework provides the functionality for a v22 application to control and install the packages required for its proper execution.

	Paquet	***Commentaire***
LibcURL	libcurl4	LibcURL – v22.Crl

	Paquet	Commentaire
	libcurl4-openssl-dev	LibcURL development files witj OpenSSL
MySQL	libmariadb3	MariaDB database client library – v22.Sql
	libmariadb-dev	MariaDB database development files
	libmariadb-dev-compat	MariaDB Connector/C, compatibility symlink avec MySQL
	mariadb-client	Optionnel, pour le client CLI MySQL
	mariadb-server	Optionnel, pour avoir le serveur en local sur sa station
SQLite	libsqlite3-0	SQLite 3 shared library
	libsqlite3-dev	SQLite 3 development files
TLS/https	libgnutls30	GNU TLS library - main runtime library – v22.Gui
Gtk	pkg-config	Manage compile and link flags for libraries
	libwebkit2gtk-4.0-37	Web content engine library for GTK
	libwebkit2gtk-4.0-dev	Web content engine library for GTK - development files
	libgtk-3-0	GTK graphical user interface library
	libgtk-3-dev	Development files for the GTK library
	libjavascriptcoregtk-4.0-dev	JavaScript engine from WebKitGTK - development files

2 Installation

2.1 Get components

Get AIDE at https://github.com/sowebio/aide-bin and choose the GNAT FSF 1124 installation which comes with latests v22 compatible Gnat compiler and GnatStudio IDE.

<<<TODO>>> Update AIDE this way, validate and restore aide-bin github repository.

Get v22 at https://github.com/sowebio/v22 and v22 documentation at https://github.com/sowebio/v22-doc

2.2 Directories

v22 comes with some inner directories: **<<<TODO>>>**

Packages	Description
bin	test binary place, with dontdelete.me test file for trailing comments preservation
doc	place of sow - v22 Ada Library User Manual.pdf and others documentation files
doc-generated	API doc generated by GNATStrudio with GNATDoc
obj/debug obj/fast obj/small obj/style	build directories
src	sources of v22
src/sys	specials system files as s-memory.adb, the GNATColl memory monitory hook
src-testapi	v22 api test program
src-testgui	v22 gui test program

2.3 Build framework and test programs

Assuming you wish to install v22 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/v22
user@system: cd v22
user@system: gprbuild -P v22
user@system: cd bin

user@system: ./testapi

user@system: ./testgui
```

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 GNAT/GCC family. GNAT FSF 1124 from AIDE is recommended.
A GNU/Linux Debian or Debian based like Ubuntu or Mint is recommended.

3 Coding guidelines

3.1 General

Language: English
Source code length: 79 columns is not mandatory anymore with our large displays.
Naming: Capitalize and user underscore with compound name. ex: Entry_Value

3.2 Messages

- Msg.Std ("Blahblah.")

  Information messages starts with a capital and ends with a dot. Ending message with three dots are only allowed when a user input is waited.

- Msg.Err ("v22.Fls.Function_Name - Can't do something.")

  Error messages starts with the library or program hierarchy following 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 v22 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

v22 is designed as a KISS working library. It does not attempt to reproduce the outstanding granularity of the Ada runtime but tries to reuse the best existing libre software ressources.

<<<TODO>>>

4.1 Types

Name	***Packages***	***Description***
Character	Standard	
String	UXString	Unbounded UTF-8 string subtyping from UXString by Pascal Pignard
Integer	Standard	
Boolean	Standard	
BCD		Financial computing
Float		Scientific computing
Geo		Geo. Coords.
	handling ok	

4.2 Packages

Name	***Packages***	***Description***
v22	Base	
Bio	Binary I/O	Binary IO: Binary files, locking, etc.
Cfg	Configuration files	Simple and user friendly config files handling
Crl	cURL interface	
Eml	Email	Pop3/Smtp
Fls	File system	
Gui	Gnoga User Interface	Gnoga high level framework
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
Usb	Usb handling	Tx, Rx and spying
Uxs	UXStrings add-ons	UXStrings functions and procedures not in UXStrings package

Name	Packages	Description
	Already coded	

4.3 Functions

About strings, v22 functions always handles and return String (UXString).

4.4 Databases

v22 handles MySQL and SQLite.

4.5 Exceptions

V22 comes with an extended post-mortem exception handler. This displays the trace not only on the screen, but also in a file bearing the application's name and .err extension.

A wide range of information is displayed and recorded:

```
--------------------------------------------------------------------------------

Exception time         : 20230911-125838
Program uptime         :  0h0 0m0 0s
Program build DT stamp : build 2023-09-11 12:58:29
Program name & version : testapi v0.7
Library name & version : v22 v0.1
Start directory        : /home/sr/opt/v22/tests
Home directory         : /home/sr
Ada mem. alloc. [bytes]: Ada Cur: [ 11396 ] Max: [ 11972 ]
All mem. alloc. [bytes]: All Cur: [ 14680064 ] Max: [ 14680064 ]

raised V22.RAISE_EXCEPTION.V22_EXCEPTION_TEST : v22.adb:43
[./testapi]
0x41d974 v22__raise_exception at v22.adb:43
0x525815 testapi_sql__run at testapi_sql.adb:512
0x4153b4 _ada_testapi at testapi.adb:179
0x41c6d1 main at b__testapi.adb:657
[/lib/x86_64-linux-gnu/libc.so.6]
0x7fd4d4a29d8e
0x7fd4d4a29e3e
[./testapi]
0x413353 _start at ???
0xfffffffffffffffe


--------------------------------------------------------------------------------

Finalizations

20230911-144939 - SQL T1  - MSG - Closing SQLITE database: v22_testapi_1
20230911-144939 - SQL T1  - MSG - Closing SQLITE database: v22_testapi_2

--------------------------------------------------------------------------------
```

When an exception occurs, the databases connections are cleanly closed, the console text cursor and specific exit status are set.

- Implementation notes

  In the GNAT Pro and CE releases, AdaCore has integrated the GNU/GCC utility addr2-line into the GNAT runtime. Addr2line is called to display the subroutine names and line numbers of the call stack.

  This feature is missing in the GNAT FSF version. This is a recurring complaint on the net, some people offer workarounds, but none of themwork here [Linux, GNAT FSF 11]. Since the CE versions are terminated, there were two ways to restore this useful feature; the clean one, rebuild a full FSF compiler with this feature, and the fast dirty one. We obviously choose the latter.

  When handling an exception, v22 search if the addr2line utility exists on the system and then, for each line indicating a "???", we run, on the executable itself, a command addr2line -e <executable name> <address> and analyze the return. If the latter is valid [example: /home/sr/.../debug/gnx-utl_sv.adb:1174, as opposed to "addr2line: DWARF error: can't find .debug_ranges section." and/or "??:?", ??:0"], the function replaces "???" by "gnx-utl_sv.adb:1174".

  If the program was compiled with GNAT Pro or CE, this is transparent and if add2line was not found, the "???" are left as they are.

5 Components

5.1 Gnoga

Excerpts from ~/Gnoga/doc/user_guide.md by David Botton.

5.1.1 What is Gnoga

Defining Gnoga is an important first step to using it. Gnoga is a framework and associated tools for developing GUI applications using the Ada language, leveraging web technologies to provide a rendering engine. Gnoga should not be confused with a web development framework. While Gnoga is very capable of creating web applications and perform in that role and is even more capable than most web development frameworks, using Gnoga for web applications is only one possibility. Native applications for desktop and mobile are just as easy to create, all using the same code base.

5.1.2 How does Gnoga work?

A Gnoga application can be divided into three parts:
- The application code written in Ada using the Gnoga framework and tools;
- The communication layer;
- The GUI rendering surface, an HTML-5 compliant browser or an embedded widget for native platform development.

The communication layer is not passive as in typical web programming using HTTP, nor is it using Ajax calls to simulate a live-active connection. It is an active, open connection using HTML5 web sockets or direct access at the API level to an embedded widget.

Since the communication layer is always active there is a stateful constant connection to the rendering surface as there is in traditional desktop GUI development even if the rendering surface is a remote browser.

The idea of using publishing technologies to display a GUI is not new. Next used Postscript; Mac OS X PDF; and Gnoga uses HTML 5.

5.1.3 Singleton and multi-connect applications

There are two basic application types in Gnoga, Singleton applications and Multi Connect applications. Singleton applications are ideal for single user desktop use. They allow only a single connection and exit when that connection is lost. Since the application will not be accessed in parallel by other connections implementation is easier in that there is no need to protect data except from parallel incoming events from a single user (In Gnoga events are not serialized and are fired as they are generated concurrently).

5.1.4 Concurrency and exceptions

While this example works, there is an issue. In Gnoga and in particular in Multi-Connect applications, concurrency and exceptions must always be considered, given that:

- While highly unlikely, it is possible for our Users vector to be accessed concurrently, something that that standard Ada containers are not designed to handle. If this were a production application, Users should be protected;

- While again highly unlikely given our application, it is possible for a view to be destroyed during the On_Change event. This could result in trying to call User_View.Draw on an already deallocated object. Therefore, it would be a good idea to capture exceptions in the On_Change event at the very least, or as part of protecting the Users vector a means be included to insure the validity of the User_View before calling Draw.

Gnoga will handle most exceptional situations not handled in your code, but creating a solid Multi-Connect application should include considerations for the above in all designs. Improving on our example is left as an exercise to the reader.

5.1.5 Advanced: The "Connection" Parameter and GUI elements on the Stack

The extra parameter "Connection" in our controller procedure "Default" can be used when you wish to block the connection procedure until the connection is closed. The two common uses of Connection.Hold to block until connection loss are:

- To add clean up code on connection loss to the connection procedure; this could also have been added to the On_Destroy event for Main_Window.

- To prevent finalization of statically defined GUI elements within the connection procedure until the connection has been lost.

An example of this second method would allow us to rewrite the skeleton procedure as:

```
``` ada
   procedure Default
     (Main_Window : in out Gnoga.Gui.Window.Window_Type'Class;
      Connection  : access
        Gnoga.Application.Multi_Connect.Connection_Holder_Type)
```
```

```
   is
      View : GnogaBoard.View.Default_View_Type;
   begin
      View.Create (Main_Window);
      View.Click_Button.On_Click_Handler (On_Click'Access);

      Connection.Hold;
   end Default;
```

5.1.6 Advanced: Per Connection App Data

In the multi-connect example above we use the connection's main view to store data specific for each user connection. It is often convenient to have a data structure containing the data specific to a connection. Gnoga offers a way to associate data to a connection and allow access to those data through any GUI element on that connection.

The following is an example:

```ada
   type App_Data is new Connection_Data_Type with
      record
         Main_Window : Window.Pointer_To_Window_Class;
         Hello_World : aliased Common.DIV_Type;
      end record;
   type App_Access is access all App_Data;

   procedure On_Click (Object : in out Gnoga.Gui.Base.Base_Type'Class;
                       Event  : in     Gnoga.Gui.Base.Mouse_Event_Record)
   is
      App : App_Access := App_Access (Object.Connection_Data);
   begin
      App.Hello_World.Text ("I've been clicked");
   end On_Click;

   procedure On_Connect
     (Main_Window : in out Gnoga.Gui.Window.Window_Type'Class;
      Connection  : access
        Gnoga.Application.Multi_Connect.Connection_Holder_Type)
   is
      App : App_Access := new App_Data;
   begin
      Main_Window.Connection_Data (App);
      App.Main_Window := Main_Window'Unchecked_Access;
      App.Hello_World.Create (Main_Window, "Click Me");
      --  By default Connection_Data is assumed to be a dynamic object
      --  and freed when the connection is closed.  To use static app
      --  data pass Dynamic => False
      ...
   end On_Connect;
```

5.1.7 Multi Connect Applications for a Single User

A Multi-Connect application allows multiple connections to the same application at the same time. This does not always imply multiple users, it could even be the same user with multiple browser windows connected to the same application. When using a multi-connect application as a single-user desktop application you simply need to re-

strict access to the application to the local machine and provide some way for the application to know it is time to shutdown. Some tips:

- In the On_Connect procedure check if there has already been a connection and if a reconnect is tried return a view that indicates application is already running.

- If limiting to only one main window, use that window's On_Destroy event to tell the application to shut down using Gnoga.Application.Multi_Connect.End_Application if you do not provide some other way to exit the application, or track connections and end the application when all connections are closed.

- Limit connections to the local machine only: in initialize use Initialize (Host => "127.0.0.1");

5.1.8 Gnoga Types

By convention in Gnoga all types are usually defined in the following way and using the following naming convention:

```
type Some_Type is ....;
type Some_Access is access all Some_Type;
```

if Some_Type is a tagged type:

```
type Pointer_to_Some_Class is access all Some_Type'Class
```

5.1.9 Directory structure when developing apps

If you use the gnoga_make tool it will set up a development directory structure in addition to creating a skeleton application. (See the Singleton and Multi-Connect examples). For reference the following directory structure is the basic structure:

```
App Dir
   |
   |___ bin - your gnoga app binary
   |
   |___ html - boot.html (or other boot loader used)
   |
   |___ js - must contain jquery.min.js and boot.js
   |
   |___ css - optional, all files served as CSS files
   |
   |___ img - optional, files served as graphics files
   |
   |___ src - Source code for your gnoga app
   |
   |___ obj - Build objects
   |
   |___ templates - optional, if using Gnoga.Server.Template_Parser
   |
   |___ upload - optional, optional directory for incoming files
```

If any of the subdirectories are missing, the files expected to be in them are assumed to be in html. If the html subdirectory is also missing these files are assumed to be in App Dir. The executable can be in the bin directory or in App Dir.

5.1.10 Application, Types, Gui, Server, Client

See Gnoga API Application, Types, Gui, Server, Client for more details.

5.1.11 Plugins, Modules

Users can write and publish to the Gnoga Marketplace two Gnoga-specific UI extension types, Plugins and Modules.

Plugins, including jQuery, jQueryUI, Boot_Strap, and Ace_Editor, are bindings to JavaScript libraries for use on the client side.

Modules are unique Gnoga-based UI elements written with Gnoga.

5.1.12 Tags Bound in Gnoga

While Gnoga is not exactly HTML in Ada, knowing the relationships may be of assistance in developing your application:

- HTML5 Tags Bound as Gui Elements in Gnoga

```
<a>, <hr>, <br>, <button>, <div>, <img>, <meter>, <progress>, <p>
        -> Element.Common,  also see Gui.View for <div> and <span>
<canvas>
        -> Element.Canvas
<svg>
        -> Element.SVG
   <form>, <input>, <textarea>, <select>, <datalist>, <legend>, <label>, <option>, <optgroup>
        -> Element.Form,
<fieldset>
        -> Element.Form.Fieldset
<audio>, <video>, <source>*, <track>*
        -> Element.Multimedia,  * Not needed
<iframe>
        -> Element.IFrame
<html>, <body>, <head>
        -> Access through Window_Type.Document
<ul>, <ol>, <li>, <dl>, <dd>, <dt>
        -> Element.List
<address>, <article>, <aside>, <header>, <main>, <nav>, <p>, <pre>, <section>
        -> Element.Section
<code>, <strong>, <em>, <dfn>, <samp>, <kbd>, <var>, <marked>, <del>, <ins>, <s>, <q>, <big>, <smal
l>, <time>, <tt>, <wbr>
        -> Element.Phrase
<link>, <style>, <title>
        -> Element.Style_Block,  The Style property on Element_Type
        -> Document.Load_CSS,  Document.Title
        -> Since content is generated by code
<table>, <caption>, <td>, <tr>, <th>, <col>, <colgroup>, <tfoot>, <thead>
        -> Element.Table
```

- HTML5 Tags Unbound as Gui Elements in Gnoga

Note: All tags can be bound and used with Element_Type.Create_With_HTML. For various reasons as described here, they are not bound specifically.

```
<map>, <area>
        -> No specific bindings currently for image maps; best
        -> generated with an automated tool as regular HTML.
<bdi>, <bdo>, <ruby>, <rp>, <rt>,  <details>, <output>, <figure>, <flgcaption>
        -> Text formatting tags are not bound.  Have no application
        -> specific use.  Span_Type should be used to contain text
        -> that needs interaction or interactive styling.
<object>, <embed>, <script>, <noscript>, <param>, <applet>
        -> No bindings are made for external plugins or scripting tags
<base>, <meta>
        -> base and meta only make sense for static pages
<dialog>, <keygen>, <menu>, <menuitem>
        -> No browsers support these tags in a way worth binding yet
<frameset>, <frame>, <noframes>
        -> No window level frame support; see Element.IFrame
```

5.1.13 Gnoga Concepts: in and out of the DOM

An HTML document is a hierarchical collection of objects. In a browser window, the displayed document is the browser's DOM, but it is possible within JavaScript to have objects that are not in the main DOM and have their own hierarchical collections, DOMs.

Gnoga maintains on the browser side JavaScript references to GUI elements it creates; these elements may or may not also be in the browser's DOM. When creating a new object in Gnoga, if the parent is in the browser's DOM, the child will be there as well. If not, it will just be part of the parent's individual DOM and not be visible, and even changing the visibility of that object will not make it appear on the browser window since they are not in the Browser's DOM.

To take an object and all its children out of the DOM use Gnoga.Gui.Element.Remove; to place it back in the DOM use one of the Place_* methods in Gnoga.Gui.Element.

5.1.14 Gnoga Concepts: Display, Visible, Hidden

HTML5 uses a few different independently working properties for visibility.

Visible will turn on or off the visibility of an element and its children, but the objects will still take the same space on the page.

Hidden will turn off visibility and the object and its children will no longer take up space on the page, either.

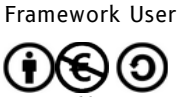
Display changes how the elements are laid out by the browser. Using Display (None) acts in the same fashion as Hidden.

5.1.15 Gnoga Concepts: Inner_HTML, Text and Value

Retrieving the contents of an Element in Gnoga differs depending on the type of Element. For form Elements the Value method is used. For others Text can be used to retrieve the text alone or Inner_HTML to retrieve the contents including any HTML tags present.

The reason there are different methods is based on the way the underlying HTML 5 works. Text and Inner_HTML are retrieving all child nodes with in the element while

Value is an attribute of Form elements. So Text or Inner_HTML will return the contents of every child.



Examples

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 Testapi

<<<TODO>>>

2 Testgui

<<<TODO>>>

3 Event listener interfaces with Gnoga

Reproduction of ~/Gnoga/doc/articles/event_listerners.txt by Jeremiah Breeden.

An option that I have used is to create event listener interface. I extend the button class to include an event generator that maintains a list of listeners and I add an Add_Listener procedure so outside views can add themselves as listeners. I create a handler that Gnoga expects but have that handler operate the generator to fire the events out of the button class.

Then I extend whichever view I am using to implement the listening interface, override the create procedure, and have it call the Add_Listener procedure mentioned above. Then I just have to override the listener procedure that handles the event and I have both the parent view that is interested in the event and the button that sourced the original event.

It's a bit complex, but it allows me to have any structure I want be able to respond to any event just by adding itself as a listener to the event.

The big thing is Handle_On_Click gives you the containing view type which allows you to modify other controls using the on click from the button. You can also pass the on click event to other parent views who may want to modify their own layouts based on that button event. You would just need to add a public Add_On_Click_Listener (or

whatever name) procedure to Listener_View.View_Type (or whatever view you create) and have it call the button's Add_On_Click_Listener procedure.

Button_For_Listeners is the extended class for Gnoga's Button_Type
Listener_View is just a test view type that would contain a Button among other controls.

button_for_listeners.ads

```
with Gnoga.Gui.Element.Common;
with Ada.Containers.Vectors;

package Button_For_Listeners is

   type Listener_Type is limited interface;

   type Button_Type is new Gnoga.Gui.Element.Common.Button_Type with private;

   procedure Add_On_Click_Listener
     (Self     : in out Button_Type;
      Listener : not null access Listener_Type'Class);

   procedure Handle_On_Click(Listener : in out Listener_Type;
                             Source   : in out Button_Type'Class) is abstract;


private
   type Listener_Class_Access is access all Listener_Type'Class;
   package Vectors is new Ada.Containers.Vectors(Natural, Listener_Class_Access);

   type Button_Type is new Gnoga.Gui.Element.Common.Button_Type with record
      On_Click_Listeners : Vectors.Vector;
   end record;

end Button_For_Listeners;
```

button_for_listeners.adb

```
with Gnoga.Gui.Base;

package body Button_For_Listeners is

   procedure Gnoga_On_Click(Object : in out Gnoga.Gui.Base.Base_Type'Class) is
      Cursor : Vectors.Cursor := Button_Type(Object).On_Click_Listeners.First;
   begin
      while Vectors.Has_Element(Cursor) loop
         Vectors.Element(Cursor).Handle_On_Click(Button_Type(Object));
         Cursor := Vectors.Next(Cursor);
      end loop;
   end Gnoga_On_Click;

   procedure Add_On_Click_Listener
     (Self     : in out Button_Type;
      Listener : not null access Listener_Type'Class) is
   begin
      Self.On_Click_Listeners.Append(Listener);
      Self.On_Click_Handler(Gnoga_On_Click'Access);
   end Add_On_Click_Listener;

end Button_For_Listeners;
```

listener_view.ads

```
with Gnoga.Gui.View;
with Gnoga.Gui.Base;
with Button_For_Listeners;

package Listener_View is
   package Buttons renames Button_For_Listeners;

   type View_Type is new Gnoga.Gui.View.View_Type
                         and Buttons.Listener_Type with private;

   overriding
   procedure Create
     (Self   : in out View_Type;
      Parent : in out Gnoga.Gui.Base.Base_Type'Class;
      ID     : in     String  := "");

private
   type Self_Ref_Type(Ref : access Buttons.Listener_Type'Class)
      is limited null record;

   type View_Type is new Gnoga.Gui.View.View_Type
                         and Buttons.Listener_Type with record
      Button : Buttons.Button_Type;
      Self_Reference : Self_Ref_Type(View_Type'Access);
   end record;

   overriding
   procedure Handle_On_Click(Listener : in out View_Type;
                             Source   : in out Buttons.Button_Type'Class);

end Listener_View;
```

listener_view.adb

```
package body Listener_View is
   procedure Create(Self   : in out View_Type;
                    Parent : in out Gnoga.Gui.Base.Base_Type'Class;
                    ID     : in     String  := "") is
   begin
      Gnoga.Gui.View.View_Type(Self).Create(Parent);
      Self.Button.Create(Self, "Test Button");
      Self.Button.Add_On_Click_Listener(Self.Self_Reference.Ref);
   end Create;

   Test_Count : Natural := 0;

   overriding
   procedure Handle_On_Click(Listener : in out View_Type;
                             Source   : in out Buttons.Button_Type'Class) is
   begin
      Listener.Button.Text("Testing: " & Integer'Image(Test_Count));
      Test_Count := Test_Count + 1;
   end Handle_On_Click;

end Listener_View;
```

4 Productivity tools

Gnoga allows you to leverage existing HTML tools to quickly design and create Forms for use in Gnoga applications.

Here is a simple step by step guide to quickly creating a bootstrap based Form using a simple online tool with Gnoga.

4.1 Form builder

Amended reproduction of ~/Gnoga/doc/articles/QuickandSimpleFormBuildingwith-Gnoga.pdf, author unknown.

4.1.1 Create form

http://bootsnipp.com/forms?version=3

Click on Form Name for the title that will be displayed above the form, for this Example: “Hello World Gnoga and Bootstrap”

Drag a “Text Input” on to the form with your mouse

Click on the Text Input and fill in:
- ID/Name : fname
- Label Text : First Name
- Placeholder : blank
- Help Text : Enter your first name
- Click required
- Leave input size at Medium
- Click Save

Drag a “Text Input” on to the form with your mouse under the First Name field

Click on the Text Input and fill in:
- ID/Name : lname
- Label Text : Last Name
- Placeholder : blank
- Help Text : Enter your last name
- Click requied
- Leave input size at Medium
- Click Save

Click on the “buttons” tab above the form elements

Drag a Single Button over under the two fields

Click on the Single Button and fill in:
- ID/Name : sbutton
- Label Text : blank
- Button Label : Ok
- Leave button type : Primary
- Click Save

4.1.2 Create Gnoga Project

Let’s create our skeleton project using gnoga_make:

- Run [add path if needed]: gnoga_make new form_demo1 multi_connect
- Edit the path [if needed] to gnoga at form_demo1/src/form_demo1.gpr

Replace boot.html copy ~/gnoga/html/boot_starp3.html to replace boot.html from the html directory of your project.

From the css directory of your project, copy the bootstrap.min.css file from the bootstrap dist directory [see below].

From the js directory of your project, copy the bootstrap.min.js file from the bootstrap dist directory [see below].

Make the directory templates:
- mkdir templates

Copy the content of Rendered or View HTML tab to the body of form_demo1.html.

Edit our view spec form_demo1view.ads:
- Add with Gnoga.Gui.Element.Form;
- Replace the record contents of Default_View Type with:
- First_Name : Gnoga.Gui.Element.Form.Text_Type;
- Last_Name : Gnoga.Gui.Element.Form.Text_Type;

Edit the view form_demo1view.adb:
- In the procedure Create after the call to the parent Create add:
- Replace the Create procedure body with:

```
Gnoga.Gui.View.View_Type [View].Create [Parent, ID];
View.Load_File ["form_demo1.html"];
View.First_Name.Attach_Using_Parent [View, "fname"];
View.Last_Name.Attach_Using_Parent [View, "lname"];
```

Edit the form_demo1controller.adb:
- Replace the On_Click with:

```
procedure On_Submit [Object : in out Gnoga.Gui.Base.Base_Type'Class];
procedure On_Submit [Object : in out Gnoga.Gui.Base.Base_Type'Class] is
View : form_demo1.View.Default_View_Type renames
form_demo1.View.Default_View_Type [Object];
begin
Gnoga.Log ["First Name : " & View.First_Name.Value];
Gnoga.Log ["Last Name : " & View.Last_Name.Value];end On_Submit;
```

- Replace the body of Default with:

```
View.Dynamic;
View.Create [Main_Window];
View.On_Submit_Handler [On_Submit'Access];
```

Now build the project by running make at the root dir of your project and open your browser to localhost:8080 and your done :]

The complete View and Controller files follow along with the form_demo1.html clip from the website:

```
with Gnoga.Gui.Base;
with Gnoga.Gui.View;
with Gnoga.Gui.Element.Form;

package form_demo1.View is

 type Default_View_Type is new Gnoga.Gui.View.View_Type with
 record
  First_Name : Gnoga.Gui.Element.Form.Text_Type;
  Last_Name : Gnoga.Gui.Element.Form.Text_Type;
 end record;

type Default_View_Access is access all Default_View_Type;
type Pointer_to_Default_View_Class is access all Default_View_Type'Class;

overriding procedure Create (View : in out Default_View_Type; Parent : in out Gnoga.Gui.Base.Base_Type'Class; ID : in String := "");

end form_demo1.View;
```

```
package body form_demo1.View is


  Create

 overridingprocedure Create (View : in out Default_View_Type; Parent : in out Gnoga.Gui.Base.Base_Type'Class; ID : in String := "") is
 begin
  Gnoga.Gui.View.View_Type (View).Create (Parent, ID);
  View.Load_File ("form_demo1.html");
  View.First_Name.Attach_Using_Parent (View, "fname");
  View.Last_Name.Attach_Using_Parent (View, "lname");
 end Create;

end form_demo1.View;
```

```
with Gnoga.Gui.Base;
with form_demo1.View;

package body form_demo1.Controller is

 procedure On_Submit (Object : in out Gnoga.Gui.Base.Base_Type'Class);
 procedure On_Submit (Object : in out Gnoga.Gui.Base.Base_Type'Class) is
  View : form_demo1.View.Default_View_Type renames
  form_demo1.View.Default_View_Type (Object);
  begin
   Gnoga.Log ("First Name : " & View.First_Name.Value);
   Gnoga.Log ("Last Name : " & View.Last_Name.Value);
  end On_Submit;

 procedure Default (Main_Window : in out Gnoga.Gui.Window.Window_Type'Class; Connection : access Gnoga.Application.Multi_Connect.Connection_Holder_Type) is
  View : form_demo1.View.Default_View_Access := new form_demo1.View.Default_View_Type;
 begin
  View.Dynamic;
  View.Create (Main_Window);
  View.On_Submit_Handler (On_Submit'Access);
 end Default; begin

 Gnoga.Application.Multi_Connect.On_Connect_Handler (Default'Access, "default");
```

```
end form_demo1.Controller;
```

4.1.3 templates/form_demo.html

```
<form class="formhorizontal">
<fieldset>

<! Form Name >
<legend>Hello World Gnoga and Bootsrap</legend>

<! Text input>
<div class="formgroup">
 <label class="colmd4 controllabel" for="fname">First Name</label>
  <div class="colmd4">
   <input id="fname" name="fname" type="text" placeholder="" class="formcontrol input-
md"required="">
  <span class="helpblock">Enter your first name</span>
 </div>
</div>

<! Text input>
<div class="formgroup">
 <label class="colmd4 controllabel" for="lname">Last Name</label>
  <div class="colmd4">
   <input id="lname" name="lname" type="text" placeholder="" class="formcontrol input-
md" required="">
   <span class="helpblock">Enter your last name</span>
 </div>
</div>

<! Button >
<div class="formgroup">
 <label class="colmd4 controllabel" for="sbutton"></label>
  <div class="colmd4">
   <button id="sbutton" name="sbutton" class="btn btnprimary">Ok</button>
  </div>
</div>

</fieldset>
</form>
```

4.1.4 Bootstrap-Form-Builder

https://github.com/minikomi/Bootstrap-Form-Builder using Bootstrap 2.3.1
http://bootsnipp.com/forms?version=3 using Bootstrap 2.3.1

4.1.5 Bootstrap library

https://getbootstrap.com

git clone https://github.com/twbs/bootstrap

git checkout tags/v2.3.2 -b v2.3.2-branch
Basculement sur la nouvelle branche 'v2.3.2-branch'

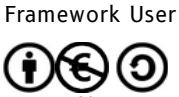
git checkout tags/v3.4.1 -b v3.4.1-branch
Basculement sur la nouvelle branche 'v3.4.1-branch'

4.2 Page builder

<<<TODO>>>

Evaluate the online page builder https://bootsnipp.com/builder.

Needs Bootstrap 4 for widgets.



API

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

1 v22

1.1 Introduction

1.1.1 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 – most of the times - these choices.

On a simple loop, let's recall that if HAC [a very valuable and helpful Ada subset interpreter] is [among others] 7 times faster and vastly more capable than Bash, HAC itself is 300 times slower than Ada. So... Loop to the grayed second line :]

1.1.2 Conventions

To ease developers:

- String type is a subtype of UXString. This new String UTF-8 subtype is the standard v22 type. You should always use String[5].

- All strings constants and function only returns String type.

- All strings parameters accept String type.

1.1.3 Usage

Use ./v22/v22.gpr as a stub for your own projects.

Use ./v22/example/ as an application template useful to create your own projects.

[5] Only use Standard.String when appropriate.

1.2 v22

Base package.

1.2.1 Get_Build

- Description

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

- Usage

  function Get_Build return String

- Example

```
Log.Msg ["Date stamp build: " & v22.Get_Build];
build 2021-08-04 14:36:27
```

1.2.2 Get_Log_Dir

- Description

  Returns the log directory.

- Usage

  function Get_Log_Dir return String

- Example

```
Log.Msg ["v22.Get_Log_Dir];
/var/log/
```

1.2.3 Get_Tmp_Dir

- Description

  Returns the temporary files directory.

- Usage

  function Get_Tmp_Dir return String

- Example

```
Log.Msg ["v22.Get_Tmp_Dir];
/tmp/
```

1.2.4 Get_Version

- Description

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

- Usage

  function Get_Version return String

- Example

```
Log.Msg ["Library version: " & v22.Get_Version];

v22 v0.6
```

1.2.5 Raise_Exception

- Description

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

  In addition to the usual trace, a v22 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/Seafile/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 ???

--------------------------------------------------------------------------------
```

1.3 Cfg - Configuration files

1.3.1 Close

- Description

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

- Usage

  procedure Close

- Example

1.3.2 Comment

- Description

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

- Usage

  procedure Comment (Text : String)

- Example

1.3.3 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

1.3.4 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 String

- Example

1.3.5 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

1.3.6 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

1.4 Crl - cURL

Crl is a comprehensive binding to libcURL as cURL is an very useful tool when dealing with almost everything around internet.

General information here: https://curl.se/libcurl/c/libcurl.html.

The recommended way to talk with libcURL is the “Easy interface”: (https://curl.se/libcurl/c/libcurl-easy.html).

1.4.1 Get_Version

- Description

  Return cURL version. On Ubuntu 22.04 LTS, the standard libcurl got theses caps: libcurl/7.81.0 OpenSSL/3.0.2 zlib/1.2.11 brotli/1.0.9 zstd/1.4.8 libidn2/2.3.2 libpsl/0.21.0 (+libidn2/2.3.2) libssh/0.9.6/openssl/zlib nghttp2/1.43.0 librtmp/2.3 OpenLDAP/2.5.16

- Usage

  function Get_Version return VString;

- Example

```
Tio.Put_Line (Crl.Get_Version);

cURL version: libcurl/7.81.0 OpenSSL/3.0.2 zlib/1.2.11 brotli/1.0.9 zstd/1.4.8 libidn2/2.3.2 libpsl/0.21.0 (+libidn2/2.3.2) libssh/0.9.6/openssl/zlib nghttp2/1.43.0 librtmp/2.3 OpenLDAP/2.5.16
```

1.5 Fls - Files

1.5.1 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);

- Example

1.5.2 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)

- Example

1.5.3 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

- Example

1.5.4 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
  procedure Delete_Directory_Tree (Dir_Tree : String)

- Example

1.5.5 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)

- Example

1.5.6 Delete_Lines

- Description

  Search and remove file lines matching Pattern in File_Name.

- Usage

  procedure Delete_Lines (File_Name, Pattern : String)

- Example

1.5.7 Download_File

- Description

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

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

- Usage

  function Download_File (Url : Sting; Dlfile : VString; Name : String; Dlsize : Integer := 0) return Boolean;

- Example

1.5.8 Exists

- Description

  Returns True if file or directory Name exists.

- Usage

  function Exists (Name : String) return Boolean

- Example

```
if Fls.Exists (HAC_Dir & "/hac") then
   Tio.Put_Line ("HAC installation is done :)");
end if;
```

1.5.9 Extract_Directory

- Description

  Returns directory from Name.

- Usage

  function Extract_Directory (Name : String) return String

- Example

```
Tio.Put_Line (Extract_Directory ("/etc/ssh/sshd_config"))

/etc/ssh
```

1.5.10 Extract_Name

- Description

  Returns filename from Name.

- Usage

  function Extract_Filename (Name : String) return VString

- Example

```
Tio.Put_Line (Fls.Extract_Filename ("/etc/ssh/sshd_config")) then

sshd_config
```

1.5.11 File_Size

- Description

  Return size of Name file.

- Usage

  function File_Size (Name : String) return Integer

- Example

1.5.12 Get_Directory

- Description

  Returns current directory.

- Usage

  function Current_Directory return String

- Example

1.5.13 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

- Example

```
Tio.Put_Line (Fls.Is_Root_Directory ("/etc"));
True

Tio.Put_Line (Fls.Is_Root_Directory ("/etc/network"));
False
```

1.5.14 Move_File

- Description

  Move 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 Move_File (Source_Name, Target_Name : String)
  ```

- Example

1.5.15 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)
  ```

- Example

1.5.16 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
  ```

- Example

1.5.17 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

- Example

1.6 Msg – Logging

To avoid name conflicts with Gnoga, which has a log procedure, the original v20.Log package has been renamed v22.Msg. Then original procedure v20.Msg has been renamed v22.Std as Standard message.

1.6.1 Dbg

- Description

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

- Usage

  procedure Dbg (Message : in String)

- Example

1.6.2 Err

- Description

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

- Usage

  procedure Err (Message : in String)

- Example

1.6.3 Get_Debug

- Description

  Return true if debug status is on.

- Usage

  function Get_Debug return Boolean

- Example

1.6.4 Get_Dir

- Description

  Returns log file directory.

- Usage

  function Get_Dir return String

- Example

1.6.5 Line

- Description

  Log a blank line.

- Usage

  procedure Line

- Example

1.6.6 Set_Debug

- Description

  Set debug messages status on/[off].

- Usage

  procedure Set_Debug [Action : Boolean]

- Example

1.6.7 Set_Dir

- Description

  Set log file directory.

- Usage

  procedure Set_Dir (Dir_In : String)

- Example

1.6.8 Set_Display

- Description

  Log to display on/[off].

- Usage

  procedure Set_Display (Action : Boolean)

- Example

1.6.9 Set_Disk

- Description

  Log to disk on/[off].

- Usage

  procedure Set_Disk (Action : Boolean)

- Example

1.6.10 Set_Header

- Description

  Line header on/[off].

- Usage

  procedure Set_Header (Action : Boolean)

- Example

1.6.11 Set_Task

- Description

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

- Usage

  function Log_Dir return String

- Example

1.6.12 Std

- Description

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

- Usage

```
procedure Std (Message : Boolean)
procedure Std (Message : ASCII_Character)
procedure Std (Message : String)
procedure Std (Message : Integer)
procedure Std (Message : Long_Integer)
procedure Std (Message : Money)
```

- Example

1.6.13 Title

- Description

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

- Usage

  procedure Title (Message : in String);

- Example

1.7 Net - Network

1.7.1 Command

- Description

  Send remote command to host. Returns True if command successful (remote exitcode equal to 0). If used, SE_Output returns remote console output.

- Usage

  function Command (Target : in String ; Command_In : in String ; SE_Output : out String) return Boolean
  function Command (Target : in String ; Command : in String) return Boolean
  procedure Command (Target : in String ; Command : in String)

- Exception

Error_Command	Raised when send command error

- Example

```
List files in a directory:

Net.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");
```

1.7.2 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 String ; File_Tx : in String; Directory_Rx : in String ; Options : in String := +"") return Boolean;
  procedure Copy_File (Target : in String ; File_Tx : in String; Directory_Rx : in String ; Options : in String := +"");

- 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 recusrsive copy Will
transfer ./usr/* to n203c1:/etc/xen-tools/skel/usr/*
```

1.7.3 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 String ; Dir_Tree : String) return Boolean

- Example

1.7.4 Delete_File

- Description

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

- Usage

  function Delete_File (Target : in String ; File_To_Delete : in String) return Boolean;
  procedure Delete_File (Target : in String ; File_To_Delete : in String);

- 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");
```

1.7.5 Directory_Exists

- Description

  Returns True if distant directory Name exists.

- Usage

  function Directory_Exists (Target : in String, Name : String) return Boolean

- Example

```
if Net.Directory_Exists (+"host", "Directory_to_test") then
   Tio.Put_Line ("Directory_to_test exists");
end if;
```

1.7.6 File_Exists

- Description

  Returns True if file Name exists.

- Usage

  function File_Exists (Target : in String, Name : String) return Boolean

- Example

```
if Net.File_Exists (+"host", "filename_to_test") then
   Tio.Put_Line ("filename_to_test exists");
end if;
```

1.7.7 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 String;

- Example

```
Tio.Put_Line ( Net.Get_Network_From_Ip ("120.1.1.1"));;
120.1.1
Tio.Put_Line ( Net.Get_Network_From_Ip ("320.1.1.1"));;
Empty string
```

1.7.8 Is_Ip_Ok

- Description

  IP validation

- Usage

  function Is_Ip_Ok (IP : in String) 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
```

1.7.9 Is_Ping_Ok

- Description

  Return true if target answer to a ping.

- Usage

  function Is_Ping_Ok (Target : in String) return Boolean;

- Example

```
Net.Is_Ping_Ok ("This_host_exists");
```

```
True

Net.Is_Ping_Ok ("This_host_don't_exist");

False
```

1.7.10 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 : String) return Boolean

- Example

```
Tio.Put_Line (Net.Is_Root_Directory ("/etc"));
True

Tio.Put_Line (Net.Is_Root_Directory ("/etc/network"));
False
```

1.7.11 Is_Ssh_Ok

- Description

  Return true if target answer to a ping.

- Usage

  function Is_Ssh_Ok (Target : in String) 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
```

1.7.12 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 : String];

- 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
```

1.7.13 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 : String ; Target_To_Mount : String ; Mount_Point : String ; Mount_Options : in String := +""] return Boolean
  procedure Mount_Remote [Remote_Host : String ; Target_To_Mount : String ; Mount_Point : String ; Mount_Options : in String := +""];

- Example

```
if Net.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;
```

1.7.14 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;
```

1.7.15 Set_Hostname

- Description

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

- Usage

  function Set_Hostname (Target : String ; Hostname : String) return Boolean

- Example

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

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

1.7.16 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 : String := "") 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;
```

1.7.17 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.
Net.Set_Message (False);
```

1.7.18 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.
Net.Set_Output (False);
```

1.7.19 Unmount

- Description

  Unmount a mount point on a remote host.

  The local mountpoint directory is deleted.

- Usage

  procedure Unmount (Target : String);

- Exception

Error_Unmount	Raised when unmount error

- Example

```
Local admin case: Net.Unmount ("root@i51c1.domain.tld");
Unmounts /mnt/root@i51c1.domain.tld
Local <user> case (home is /home/user): Net.Unmount ("root@i51c1.domain.tld");
```

```
Unmounts /home/<user>/mnt/sr@i51c1.domain.tld
```

1.7.20 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 : String ; Mount_Point : String) return Boolean
  procedure Unmount_Remote (Remote_Host : String ; Mount_Point : String);

- Example

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

1.8 Prg - Program

1.8.1 Command

- Description

  Constant storing program command (Arg 0).

- Usage

  Command : constant String

- Example

```
Tio_Line (Command);

/home/sr/Seafile/Sowebio/informatique/dev/ada/app/gnx/src/gnx-instance
```

1.8.2 Current_Time_Seconds

- Description

  Returns a duration as seconds since ISO date 197001010. 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_String [Current_Time_Seconds]];
1646227335
```

1.8.3 Duration_Stamp

- Description

  Returns a duration as HHhMMmSSs since Time.

- Usage

  function Duration_Stamp [Time : Ada.Calendar.Time] return String

- Example

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

1.8.4 Duration_Stamp_Seconds

- Description

  Returns a duration as seconds since Time.

- Usage

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

- Example

```

```

1.8.5 Duration_Stamp_Time

- Description

  Returns a formatted HHhMMmSSs String from Time_Seconds.

- Usage

  function Duration_Stamp_Time [Time_Seconds : Integer] return String

- Example

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

1.8.6 Generate_Password

- Description

  Password generation with 64 charset [[A-Z] + [a-z] + [0-9] + '_' + '-']
  Search space size greater than 1,26 x 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 String

- Example

```
Log.Msg [Prg.Generate_Password];

5fx7_0Fubo-hNa
```

1.8.7 Get_Version

- Description

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

- Usage

  function Get_Version return String

- Example

```
Log.Msg ["Program version: " & Prg.Get_Version];

Program version: v2.16
```

1.8.8 Get_Version_Major

- Description

  Returns Major version.

- Usage

  function Get_Version_Major return Natural;

- Example

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

```
2
```

1.8.9 Get_Version_Minor

- Description

  Returns Minor version.

- Usage

  function Get_Version_Minor return Natural;

- Example

```
Log.Msg (Prg.Get_Version_Minor);  P
16
```

1.8.10 Is_User_Not_Root

- Description

  Returns true if program user's not root.

- Usage

  function Is_User_Not_Root return Boolean

- Example

```

```

1.8.11 Name

- Description

  Return program name.

- Usage

  function Name return String

- Example

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

1.8.12 Path

- Description

  Return program path.

- Usage

  function Path return String

- Example

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

1.8.13 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

1.8.14 Set_Version

- Description

  Set program version.

- Usage

  procedure Set_Version (Major : Natural; Minor : Natural)

- Example

1.8.15 Start_Dir

- Description

  Constant storing current directory at start.

- Usage

  Start_Dir : constant String

- Example

1.8.16 Start_Time

- Description

  Constant storing Time at program start.

- Usage

  Start_Time : constant Ada.Calendar.Time

- Example

1.8.17 Time_Stamp

- Description

  Returns current timestamp as YYYYMMDD-HHMMSS

- Usage

  function Time_Stamp return String

- Example

1.9 Sql

1.9.1 General notes

This SQL package is a high-level API to simplify development. It includes, among others, the following features:

- Redundancy-free integration with low-level binding Gnoga.Server.Database;
- Standardized interface for all target databases;
- Database version management with table, index, relationship and field creation, followed by automatic update;
- Integrated data dictionary in Sys_Schema table;
- Set and Get configuration parameters in Sys_Config dedicated table;
- General purpose SQL API to ease the programmer.

See testapi demo program for further details.

Common types between MariaDB and SQLite used in v22

```
MariaDB    SQLite    Compatibility notes
-------------------------------------------------------
DECIMAL    DECIMAL   DECIMAL SQLite affinity NUMERIC
FLOAT      FLOAT     FLOAT SQLite affinity REAL
INTEGER    INTEGER   -2147483648 to 2147483647 [signed]
TEXT       TEXT      65535 max UTF-8, UTF-16BE, UTF-16LE
```

BLOB of any sizes should be store as regular files.
MONEY should be provided as **type** Money **is delta** 0.01 **digits** 15 and stored in INTEGER at the lowest unit, usually cents for euros or dollars

1.9.2 MySQL notes

<<<TODO>>>

1.9.3 SQLite notes

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

To ease customization, SQLite DB is directly compiled and linked against sqlite3.c source located in src/sql.

1.9.4 v20 to v22 transition

List of obsolete v20 functions, superseded by Gnoga.Server.Database functions.

- Column_Integer, Column_Text

  Use Gnoga.Server.Database.Field_Value and Gnoga.Server.Database.Field_Decimals.

- Column_Count

  Use Gnoga.Server.Database.Number_Of_Fields.

- Column_Type

  Use Gnoga.Server.Database.Field_Type.

- Error

  Use Gnoga.Server.Database.Error_Message.

- Exec

  Use Gnoga.Server.Database.Execute_Query.

- Last_Insert_RowID

  Use Gnoga.Server.Database.Insert_ID.Insert_ID.

- Open

  Use Gnoga.Server.Connect.

- Prepare

  Use Gnoga.Server.Query.

- Reset

  Use Gnoga.Server.Close.

- Step

  Use Gnoga.Server.Next.

1.9.5 Close

- Description

  Close a database or all database(s) when no database object specified.

- Usage

  procedure Close (DB : in out GSD.Connection'Class);
  procedure Close;

- Example

```
Sql.Close;

20230911-144939 - SQL T1  - MSG - Closing SQLITE database: v22_testapi_1
20230911-144939 - SQL T1  - MSG - Closing SQLITE database: v22_testapi_2
```

1.9.6 Column_Exists

- Description

  Return true if Column_Name exists in Table_Name.
  Return False if Column_Name or Table_Name does not exist.

- Usage

  function Column_Exists (DB : in out GSD.Connection'Class;
  Table_Name : String;
  Column_Name : String) return Boolean

- Example

```
Tio.Put ("Column_Exists: ");
Tio.Put_Line (Column_Exists (DB, "test_table", "Existing_Column")); -- Existing column

Column_Exists: True
```

```
Tio.Put ["Column_Exists: "];
Tio.Put_Line [Column_Exists [DB, "test_table", "azeazeaze"]]; -- Non existing column

Column_Exists: False
```

1.9.7 Delete

- Description

  Delete a row in Table_Name specifying a Where_Condition.

- Usage

  ```
  procedure Delete [DB : in out GSD.Connection'Class;
                    Table_Name : String;
                    Where_Condition : String]
  ```

- Example

```
--  Delete row for Number = 1234 in table Cluster

Sql.Delete [+"Cluster", "Number = 1234"];
```

1.9.8 From_Money

- Description

  Convert type Money to Integer for accurate storage.

- Usage

  ```
  function From_Money [DB_Money : Money] return Integer
  ```

- Example

```
--  Delete row for Number = 1234 in table Cluster

Sql.Delete [+"Cluster", "Number = 1234"];
```

1.9.9 Get_Config

- Description

  Get configuration Value from Parameter stored in Sys_Config table.
  The Sys_Config table must be already created.
  Returns an empty string if the Sys_Config table or parameter does not exist

- Usage

  ```
  function Get_Config [DB : in out GSD.Connection'Class;
                       Parameter : String] return String
  ```

- Example

```
-- Get parameter´s value ‘Schema_Version´ [previously set to ‘0.1´]
Sql.Get_Config [“Schema_Version”];
0.1
```

1.9.10 Get_Database_Brand

- Description

  Returns the Database_Type of DB.

- Usage

  function Get_Database_Brand [DB : in out GSD.Connection'Class]
  return Database_Brand

  type Database_Brand is [None, MySQL, SQLite, Unknown]

- Example

1.9.11 Get_Version

- Description

  Returns MySQL or SQLite database version.

- Usage

  function Get_Version [DB : in out GSD.Connection'Class] return String

- Example

```
Tio.Put_Line [DB, Sql.Get_Version [MySQL_Handle]];
MySQL: v10.3.39-MariaDB-0+deb10u1
Tio.Put_Line [DB, Sql.Get_Version [SQLiTE_Handle]];
SQLite: v3.43.0
```

1.9.12 Index_Exists

- Description

  Return true if Index_Name exists for Table_Name.
  Return False if Index_Name or Table_Name does not exist.
  Names are case insensitive for MySQL and case sensitive for SQLite.

- Usage

  function Index_Exists (DB : in out GSD.Connection'Class;
  Table_Name : String;
  Index_Name : String) return Boolean

- Example

```
Tio.Put("Index_Exists: ");
Tio.Put_Line (Sql.Index_Exists (+"key")); -- Existing index

Tio.Put("Index_Exists: ");
Tio.Put_Line (Sql.Column_Exists (+"key1")); -- Non existing index

…

Index_Exists: True
Index_Exists: False
```

1.9.13 Insert

- Description

  Create a row in Table_Name with Columns_Values.

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

  A non existent Table or Column don't raise exception but an error is logged.

- Usage

  procedure Insert (DB : in out GSD.Connection'Class;
  Table_Name : String;
  Columns_Values : String)

- Example

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

Sql.Insert (DB, "Cluster", "Number~1234" & "^" & "Domain~genesix.org")
```

1.9.14 Last_RowID

- Description

  Returns last existing RowID in Table_Name or 0 if Table_Name does not exist.

- Usage

  function Last_RowID (DB : in out GSD.Connection'Class;
  Table_Name : String) return Integer

- Example

```
Tio.Put_Line (Sql.Row_Count (DB, "Table_test"));
12
```

1.9.15 Open

- Description

  Open or create a database, passing a schema version for automatic schema update.

  If schema version is upper than the version stored in Database, returns Open_Need_Update as a database schema update is needed.

  URI conforms to RFC 3986 URI. See examples below:

```
MySQL:db:db_name?host=192.168.0.243&port=3306&user=user_name&password=user_password
SQLite: file:data.db or file:data.db?mode=ro&cache=private
see https://www.sqlite.org/c3ref/open.html for more information.
```

  DB_Version is major.minor format.

  Returns Database_Status.

  Schema_Load listing and Schema_Update are therefore only launched when necessary.

  To update the database schema in table Cluster with a new column named 'Thingy':

  - Increment Sql.Open version parameter from 1.0 to 1.1;
  - Add at the appropriate place in the source: Sql.Schema_Load (Sql.Column_Name, "Thingy", "TEXT").

  See example below.

- Usage

```
function Open (DBM : in out GSD.MySQL.Connection;
               URI : String := "";
               Version : String := "") return Database_Status
function Open (DBS : in out GSD.SQLite.Connection;
               URI : String := "";
               Version : String := "") return Database_Status

type Database_Status is (None,
                         Open_Failed,
                         Open_Failed_Parameter_Invalid,
                         Open_Success,
                         Open_Need_Update);
```

- Example - Before schema update

  Version is 1.0. If database does not exist, it will be created.

```
if Sql.Open (DBS, "file:sqlite_test.db", "1.0") = Open_Need_Update then

    -- Cluster table ------------------------------------------------------
    Sql.Schema_Load (Sql.Table_Name,         "Tbl_Cluster", Comment => "Clusters table");
    Sql.Schema_Load (Sql.Column_Name,        "Number", "INTEGER", "Cluster number (1) 1...240");
    Sql.Schema_Load (Sql.Column_Constraint, "Number", "UNIQUE");
    Sql.Schema_Load (Sql.Table_Constraint,   "Number", "PRIMARY KEY");

    Sql.Schema_Load (Sql.Column_Name,        "Key_Name",             "TEXT",    "Cluster key name");
    Sql.Schema_Load (Sql.Column_Name,        "Key_Private",          "BLOB",    "Cluster private key");
    Sql.Schema_Load (Sql.Column_Name,        "Key_Public",           "BLOB",    "Cluster public key");
    Sql.Schema_Load (Sql.Column_Name,        "Supervisor_Instance", "INTEGER", "Supervisor instance");
    Sql.Schema_Load (Sql.Column_Constraint, "Supervisor_Instance", "REFERENCES Tbl_Instance(Number)");
    Sql.Schema_Load (Sql.Column_Name,        "Comment",              "TEXT",    "Comment");

    Sql.Schema_Load (Sql.Index_Name,         "Idx_Cluster_Number", "Number");
    Sql.Schema_Load (Sql.Index_Constraint,   "Idx_Cluster_Number", "UNIQUE");

    Sql.Schema_Update (DBS);

end if;
```

Results:

```
Database sqlite_test needs creation or update
Create Table: Sys_Config - Create Column: Parameter TEXT UNIQUE PRIMARY KEY
CREATE TABLE Sys_Config (Parameter TEXT UNIQUE PRIMARY KEY)
Table: Sys_Config - Create Column: Value TEXT
Table: Sys_Config - Creating Index: Idx_Config_Parameter Parameter
Create Table: Sys_Schema - Create Column: Table_Name TEXT
CREATE TABLE Sys_Schema (Table_Name TEXT  )
Table: Sys_Schema - Create Column: Column_Name TEXT
Table: Sys_Schema - Create Column: Column_Type TEXT
Table: Sys_Schema - Create Column: Column_Constraint TEXT
Table: Sys_Schema - Create Column: Version TEXT
Table: Sys_Schema - Create Column: Comment TEXT
Create Table: Tbl_Cluster - Create Column: Number INTEGER UNIQUE PRIMARY KEY
CREATE TABLE Tbl_Cluster (Number INTEGER UNIQUE PRIMARY KEY)
Table: Tbl_Cluster - Create Column: Key_Name TEXT
Table: Tbl_Cluster - Create Column: Key_Private BLOB
Table: Tbl_Cluster - Create Column: Key_Public BLOB
Table: Tbl_Cluster - Create Column: Supervisor_Instance INTEGER REFERENCES Tbl_Instance(number)
Table: Tbl_Cluster - Create Column: Comment TEXT
Table: Tbl_Cluster - Creating Index: Idx_Cluster_Number Number UNIQUE
```

Tbl_Cluster	Number	INTEGER	UNIQUE	+1.0	Cluster number (1) 1...240
Tbl_Cluster	Key_Name	TEXT		+1.0	Cluster key name
Tbl_Cluster	Key_Private	BLOB		+1.0	Cluster private key
Tbl_Cluster	Key_Public	BLOB		+1.0	Cluster public key
Tbl_Cluster	Supervisor_Instance	INTEGER	REFERENCES Tbl_Instance(Number)	+1.0	Supervisor instance
Tbl_Cluster	Comment	TEXT		+1.0	Comment

- Example - A schema update is needed

  Version is upgraded to 1.1 and we add the Thingy1 and Thingy2 fields, in bold below:

```
if Sql.Open (DBS, "file:sqlite_test.db", "1.1") = Open_Need_Update then

    -- Cluster table ------------------------------------------------------
    Sql.Schema_Load (Sql.Table_Name,         "Tbl_Cluster", Comment => "Clusters table");
    Sql.Schema_Load (Sql.Column_Name,        "Number", "INTEGER", "Cluster number (1) 1...240");
    Sql.Schema_Load (Sql.Column_Constraint, "Number", "UNIQUE");
    Sql.Schema_Load (Sql.Table_Constraint,   "Number", "PRIMARY KEY");

    Sql.Schema_Load (Sql.Column_Name,        "Key_Name",             "TEXT",    "Cluster key name");
    Sql.Schema_Load (Sql.Column_Name,        "Key_Private",          "BLOB",    "Cluster private key");
    Sql.Schema_Load (Sql.Column_Name,        "Key_Public",           "BLOB",    "Cluster public key");
    Sql.Schema_Load (Sql.Column_Name,        "Supervisor_Instance", "INTEGER", "Supervisor instance");
    Sql.Schema_Load (Sql.Column_Constraint, "Supervisor_Instance", "REFERENCES Tbl_Instance(Number)");
    Sql.Schema_Load (Sql.Column_Name,        "Comment",              "TEXT",    "Comment");
```

```
    Sql.Schema_Load (Sql.Column_Name,        "Thingy1",              "TEXT",     "Thingy1 created at v1.1");
    Sql.Schema_Load (Sql.Column_Name,        "Thingy2",              "TEXT",     "Thingy2 created at v1.1");

    Sql.Schema_Load (Sql.Index_Name,         "Idx_Cluster_Number",  "Number");
    Sql.Schema_Load (Sql.Index_Constraint,   "Idx_Cluster_Number",  "UNIQUE");

    Sql.Schema_Update (DBS);

end if;

Results:

Database sqlite_test needs creation or update
Table:  Tbl_Cluster - Create Column:  Thingy1 TEXT
Table:  Tbl_Cluster - Create Column:  Thingy2 TEXT
```

Tbl_Cluster	Key_Name	TEXT		+1.1	Cluster key name
Tbl_Cluster	Key_Private	BLOB		+1.1	Cluster private key
Tbl_Cluster	Key_Public	BLOB		+1.1	Cluster public key
Tbl_Cluster	Supervisor_Instance	INTEGER	REFERENCES Tbl_Instance(Number)	+1.1	Supervisor instance
Tbl_Cluster	Comment	TEXT		+1.1	Comment
Tbl_Cluster	Thingy1	TEXT		+1.1	Thingy1 created at v1.1
Tbl_Cluster	Thingy2	TEXT		+1.1	Thingy2 created at v1.1

- Notes

  Although implementation reserves have been provided for, the table or column deletion function has not yet been implemented.

1.9.16 Properties

- Description

  Returns database properties record.

- Usage

  function Properties (DB_Name : String) return Database_Line

```
type Database_Line is record
   Brand : Database_Brand := None;
   Status : Database_Status := None;
   URI : String;
   Name : String;
   Host : String;
   Port : Natural;
   User : String;
   Password : String;
   File : String;
   Major : Natural;
   Minor : Natural;
end record;
```

- Example

1.9.17 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 v22 CD constant as Column delimiter and RD constant as Row delimiter.

- Usage

```
function Read (DB : in out GSD.Connection'Class;
               Table_Name : String;
               Columns : String;
               Condition : String := "") return String
```

- Example

```
Sql.Read (DB, "Cluster", "Number, Domain");
Sql.Read (DB, "Cluster", "Number, Domain", "WHERE Number = 1234");

-- Used in combination with Field_Display

Field_Display (Sql.Read ("Cluster", "Number,Domain"), CD, RD, "Cluster number, Domain
name"));

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

1.9.18 Row_Count

- Description

  Returns counted rows in Table_Name with Options.

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

- Usage

```
function Row_Count (DB : in out GSD.Connection'Class;
                    Table_Name : String;
                    Option : String := "*") return Integer
```

- Example

```
Tio.Put_Line (DB, Sql.Row_Count (DB, "Table_test"));
```

1.9.19 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 := "";
                    Comment : 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");
```

1.9.20 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 (DB : in out GSD.Connection'Class)
```

- 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;
```

1.9.21 Search

- Description

  Return True if Condition verified.

- Usage

```
function Search (DB : in out GSD.Connection'Class;
                 Table_Name : String;
                 Condition : String) return Boolean
```

- Example

```
if Sql.Search (DB, "Cluster","Number = 1234") then
   Tio.Put_Line ("Search 'Number = 1234': Found");
end if;
if not Sql.Search (DB, "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 (DB, "Cluster","Login = 'sr'") then
   Tio.Put_Line ("Search 'Login = sr': Found");
end if;

Search 'Login = sr': Found
```

1.9.22 Set_Config

- Description

  Store configuration Parameter and Value to Sys_Config table.
  The new Value will replaced the eventually existing one.
  The Sys_Config table will be created if needed.

- Usage

```
procedure Set_Config (DB : in out GSD.Connection'Class;
                      Parameter : String;
                      Value : String)
```

- Example

```
-- Set '0.1' value in parameter 'Schema_Version'
Sql.Set_Config (DB, "Schema_Version", "0.1");
```

1.9.23 Table_Exists

- Description

  Return true if Table_Name exists.

- Usage

```
function Table_Exists (DB : in out GSD.Connection'Class;
                       Table_Name : String) return Boolean
```

- Example

```
Tio.Put("Table_Exists: ");
Tio.Put_Line (Table_Exists (DB, "test_table")); -- Existing table

Table_Exists: True

Tio.Put("Table_Exists: ");
Tio.Put_Line (Table_Exists (DB, "test_table1")); -- Non existing table

Table_Exists: False
```

1.9.24 Update

- Description

  Update one or more 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.

  A non existent Table or Column don't raise exception but an error is logged.

- Usage

```
procedure Update (DB : in out GSD.Connection'Class;
                  Table_Name : String;
                  Columns_Values : String;
                  Where_Condition : String)
```

- Example

```
--  Update Domain column with genesix2.org value for Number = 1234 in table Cluster

Sql.Update (DB, "Cluster", "Domain~genesix2.org", "Number = 1234");
```

1.10 Sys - System

1.10.1 Command_Path

- Description

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

- Usage

  function Command_Path (Command_Name : String) return VString;

- Example

```
Sys.Command_Path ("mc");

/usr/bin/mc
```

1.10.2 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 ]
```

1.10.3 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

```
Sys.Get_Alloc_All;
```

```
All Cur: [ 2514944 ] Max: [ 2514944 ]
```

1.10.4 Get_Env

- Description

  Returns String value of String environment variable Name

- Usage

  function Get_Env (Name : String) return VString

- Example

```

```

1.10.5 Get_Home

- Description

  Returns HOME path without trailing slash.

- Usage

  function Get_Home return VString

- Example

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

1.10.6 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
```

1.10.7 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 String;

- Example

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

1.10.8 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, sub-version like 18.04**.6** and **LTS** string are omitted.

- Usage

  function Get_System_Version return String;

- Example

```
Sys.Get_System_Version - for system "Debian 11 GNU/Linux 11"
"11"
Sys.Get_System_Version - for system "Ubuntu 22.04 LTS"
"22.04"
```

1.10.9 Install_Packages

- Description

  Install packages for Debian, Ubuntu or derivatives distributions.

- Usage

  function Install_Packages (Packages_List : String; Host_Name : VString := +"") return Boolean;
  function Install_Packages (Packages_List : VString; Host_Name : VString := +"") return 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;
```

1.10.10 Is_Command

- Description

  Return true if command exists and reachable from path.

- Usage

  function Is_Command (Package_Name : String) return Boolean

- Example

```
if not Sys.Is_Command ("mc") then
   Log.Err ("Midnight Commander not available.");
end if;
```

1.10.11 Is_Package

- Description

  Return true if Package_Name is installed.

- Usage

  function Is_Package (Package_Name : String) return Boolean

- Example

```
if not Sys.Is_Package ("curl") then
   Log.Err ("Package Curl is missing.");
end if;
```

1.10.12 Purge_Packages

- Description

  Purge packages for Debian, Ubuntu or derivatives distributions.

- Usage

  function Purge_Packages (Packages_List : String; Host_Name : String := "") return 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;
```

1.10.13 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

```
Sys.Reset_Memory_Monitor;
```

1.10.14 Set_Env

- Description

  Set an environment variable Name.

- Usage

  procedure Set_Env (Name : String; Value : String)

- Example

1.10.15 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];
```

1.10.16 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 : String; Result : out Integer]
  procedure Shell_Execute [Command : String; Result : out Integer; Output : out String]

- 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.
```

1.11 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;
```

1.11.1 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

....!....|....!....|....!..
```

1.11.2 Ansi

- Description

  Get ANSI state for v20 display functions and procedures.

- Usage

  function Ansi return Boolean;

- Example

1.11.3 Ansi_Off

- Description

  Set ANSI state off for v20 display functions and procedures.

- Usage

  procedure Ansi_Off;

- Example

1.11.4 Ansi_On

- Description

  Set ANSI state on for v20 display functions and procedures.

- Usage

  procedure Ansi_On;

- Example

1.11.5 Beep

- Description

  Send a beep.

- Usage

  procedure Beep

- Example

1.11.6 Clear_Screen

- Description

  Clear the screen.

- Usage

  procedure Clear_Screen

- Example

```
-- Clear the screen

Clear_Screen;
```

1.11.7 Confirm_Twice

- Description

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

- Usage

  function Confirm_Twice (User_Prompt_1 : String ; User_Prompt_2 : String) return 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;
```

1.11.8 Cursor_Line_Backward

- Description

  Move the cursor backward X rows.

- Usage

  procedure Cursor_Line_Backward (X : Row)

- Example

1.11.9 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

1.11.10 Cursor_Line_ Forward

- Description

  Move the cursor forward X rows.

- Usage

  procedure Cursor_Line_Forward (X : Row)

- Example

1.11.11 Cursor_Line_ Move

- Description

  Move the cursor at the specified X,Y coordinates.

- Usage

  procedure Cursor_Move (X : Row; Y : Column)

- Example

1.11.12 Cursor_Off

- Description

  Hide the cursor console.

- Usage

  procedure Cursor_Off

- Example

```
Cursor_Off;
```

1.11.13 Cursor_On

- Description

  Display the cursor console.

- Usage

  procedure Cursor_On

- Example

```
Cursor_On;
```

1.11.14 Cursor_Restore

- Description

  Restore the previous saved cursor position.

- Usage

  procedure Cursor_Restore

- Example

1.11.15 Cursor_Save

- Description

  Save the current cursor position.

- Usage

  procedure Cursor_save

- Example

1.11.16 Line

- Description

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

- Usage

  procedure New_Line (Spacing : Positive)

- Example

1.11.17 Get_Immediate

- Description

  Get a character validated by [Enter]

- Usage

  procedure Get_Immediate (C : out Character)

- Example

```
procedure Pause is

Dummy :  Character;

begin

  Tio.Put_Line ("Press any key to continue...");
  Tio.Get_Immediate(Dummy);

end Pause;
```

1.11.18 Get_Password

- Description

  Returns a password blind typed

- Usage

  function Get_Password return String

- Example

```
Pass := Tio.Get_Password;

Password:
```

1.11.19 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;
```

1.11.20 Put

- Description

  Print to the console.

- Usage

```
procedure Put (B : Boolean)
procedure Put (C : Character) renames ATI.Put
procedure Put (V : String)
procedure Put (I : Integer)
procedure Put (I : Long_Integer)
procedure Put (M : Money)
```

- Example

```

```

1.11.21 Put_Line

- Description

  Print to the console then add a new line.

- Usage

```
procedure Put_Line (B : Boolean)
procedure Put_Line (C : Character)
procedure Put_Line (V : String)
procedure Put_Line (I : Integer)
procedure Put_Line (I : Long_Integer)
procedure Put_Line (M : Money)
```

- Example

```

```

1.12 Tio - Text files

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

1.12.1 Append

- Description

  Append a file.

File mode is “Out” (write mode).

- Usage

  procedure Append (Handle : in out File; Name : String)

- Example

```
Append (File_Tmp_Handle, “./toto”);
while not Tio.End_Of_File (File_Tmp_Handle) loop
  Tio.Put_Line (File_Tmp_Handle, "This is a new line of data");
end loop;
Close (File_Tmp_Handle);
```

1.12.2 Close

- Description

  Close a file.

- Usage

  procedure Close (Handle : in out File)

- Example

```
Tio.Open (File_Tmp_Handle, “./toto”);
while not Tio.End_Of_File (File_Tmp_Handle) loop
  Tio.Get_Line (File_Tmp_Handle, Line_Buffer);
end loop;
Tio.Close (File_Tmp_Handle);
```

1.12.3 Create

- Description

  Create a file.
  File mode is “Out” (write mode).

- Usage

  procedure Create (Handle : in out File; Name : String)

- Example

```
…/…
```

```
   File_Tmp_Handle :  Tio.File;

begin

   Tio.Create (File_Tmp_Handle,  +"./toto");

   Tio.Put_Line (File_Tmp_Handle, "Write a first line in ./toto");
   Tio.Put_Line (File_Tmp_Handle, "Write a second line in ./toto");

   Tio.Close (File_Tmp_Handle);

…/…
```

1.12.4 End_Of_Line

- Description

  Return true if end of line is reached.

- Usage

  function End_Of_Line (Handle : File) return Boolean

- Example

1.12.5 End_Of_File

- Description

  Return true if end of file is reached.

- Usage

  function End_Of_File (Handle : File) return Boolean

- Example

1.12.6 Flush

- Description

  Flush file buffer to disk.

- Usage

  procedure Flush (Handle : in File)

- Example

1.12.7 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);
```

1.12.8 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 String)

- Example

```
Tio.Create (File_Tmp_Handle, +"./toto");

while not Tio.End_Of_File (File_Tmp_Handle) loop

  Tio.Get_Line (File_Tmp_Handle, Line_Buffer);

end loop;

Tio.Close (File_Tmp_Handle);
```

1.12.9 Is_Open

- Description

  Returns true if Handle file is open.

- Usage

  function Is_Open (Handle : in File) return Boolean

- Example

1.12.10 Line

- Description

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

- Usage

  procedure New_Line (Handle : File; Spacing : Positive)

- Example

1.12.11 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 := +"");

- Example

```
…/…
   File_Tmp_Handle :  Tio.File;
begin
   Tio.Open_Conf (File_Tmp_Handle, +“./toto”, True, +"0600");
   Tio.Put_Line (File_Tmp_Handle, “Write a first line in ./toto”);
   Tio.Put_Line (File_Tmp_Handle, “Write a second line in ./toto”);
   Tio.Close (File_Tmp_Handle);
```

```
…/…
```

1.12.12 Open_Read

1.12.13 Description

Open a file. File mode is “In” (read mode).

- Usage

  procedure Open_Read (Handle : in out File; Name : String)

- Example

```
…/…
   File_Tmp_Handle :  Tio.File;
begin
   Tio.Open_Read (File_Tmp_Handle,  +“./toto”);
   while not Tio.End_Of_File (File_Tmp_Handle) loop
      Tio.Get_Line (File_Tmp_Handle,  Line_Buffer);
   end loop;
   Tio.Close (File_Tmp_Handle);
    …/…
```

1.12.14 Put

- Description

  Write to a file

- Usage

  procedure Put (Handle : File; C : Character)
  procedure Put (Handle : File; S : String)

- Example

1.12.15 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)
  ```

- Example

1.12.16 Read_File

- Description

  Read a text file File_To_Read and returning a String buffer. LF (line feed) are preserved.

- Usage

  ```
  function Read_File (File_Name : String) return String
  ```

- Example

1.12.17 Reset

- Description

  Reset the file pointer to the start of the file

- Usage

  ```
  procedure Reset (Handle : in out File)
  ```

- Example

1.12.18 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 : String, Content : String ; Permissions : String := +"")
  ```

- Example

1.13 Uxs - UXStrings

Uxs is a package extending UXString.

Null_String : Null_UXString

1.13.1 Char_Count

- Description

  Count each char in String_To_Process relative to Char_Set_Pattern.

- Usage

  function Char_Count (String_To_Process : String ; Char_Set_Pattern : String) return Integer;

- Example

```
Tio.Put_Line (Uxs.Char_Count ("alpha", "ap"));
3
```

1.13.2 Empty

- Description

  Return True if String Source is empty.

- Usage

  function Empty (Source : String) return Boolean;

- Example

```
Tio.Put_Line (Uxs.Empty (""));
True
```

1.13.3 Ends_With

- Description

  Check if String Item ends with another String Pattern.

- Usage

  function Ends_With (Item : String; Pattern : Character) return Boolean;

- Example

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

1.13.4 Field_* 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.

1.13.5 Field_By_Index

- Description

  Return a field indexed by Index_Field and delimited by Field_Delimiter.

- Usage

  function Field_By_Index [String_Input : String ; Index_Field : Integer ; Field_Delimiter : String] return String;

- Example

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

bravo
```

1.13.6 Field_By_Name

- Description

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

- Usage

  function Field_By_Name [String_Input : String ; Field_To_Search : String ; Field_Delimiter : String] return String;

- Example

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

```
bravo
```

1.13.7 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 : String ; Field_Delimiter : String) return Integer;

- Example

```
Tio.Put_Line (uxs.Field_Count ("alpha:bravo:charlie", ":"));
3
```

1.13.8 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 : String ; Items_List : string ; Field_Delimiter : String) return Boolean;

- Example

```
Tio.Put_Line (Uxs.Field_Count ("alpha,bravo,charlie", "alpha,charlie" , "","));
True
```

1.13.9 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 v22 (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 : String; Column_Delimiter : String; Row_Delimiter : String; Custom_Header : String := "");

- Example

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

```
Uxs.Field_Display (Sql.Read ("Cluster", "Number,Domain"), CD, RD, "Cluster number, Domain name");

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

1.13.10 Field_Search

- Description

  Search Field_To_Search in String_To_Process and return True if found.

- Usage

  function Field_Search (String_To_Process : String ; Field_To_Search : String ; Field_Delimiter : String) return Boolean;

- Example

```
Tio.Put_Line (Uxs.Field_Search ("alpha:bravo:charlie", "bravo", ":"));

True
```

1.13.11 Is_Numeric

- Description

  Return True if Item string is numeric.

- Usage

  function Is_Numeric (Item : in String) return Boolean;

- Example

```
tio.Put_Line (uxs.Is_Numeric ("12AZE12"));
False
tio.Put_Line (uxs.Is_Numeric ("1212"));
True
```

1.13.12 Replace_Char

- Description

  Replace all Char_In by Char_Out in String_To_Process.

- Usage

  function Replace_Char (String_To_Process : String ; Char_In : Character ; Char_Out : Character) return String;

- Example

```
Replace_Char (+ABCDEFGH", 'D', 'Z');
"ABCZEFGH"
```

1.13.13 Replace_Pattern

- Description

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

- Usage

  function Replace_Pattern (String_To_Process : String ; Pattern_In : String ; Pattern_Out : String) return String;

- Example

```
Uxs.Replace_Pattern ("ABCDEFGH", "BCD","xxyyzz");
"AxxyyzzDEFGH"
```

1.13.14 Starts_With

- Description

  Check if String Item starts with another String Pattern.

- Usage

  function Starts_With (Item : String; Pattern : Character) return Boolean;
  function Starts_With (Item : String; Pattern : String) return Boolean

- Example

```
- Check String with String pattern
if Ends_With ("package",  "pac") then
  Tio.Put_Line ("Match !");
end if;
```

1.13.15 Stript_Chars

- Description

  Stript each char in String_To_Process relative to Char_List.

- Usage

  function Stript_Chars (String_To_Process : String ; Char_List : String) return String

- Example

```
Tio.Put_Line (Uxs.Stript_Chars ("ABCDEFGH", "BDF"));

"ACEGH"
```

1.13.16 Tail_After_Match

- Description

  Extract a String from Source starting from Pattern+1 position to the end.
  If Pattern not found, return Source unchanged.

- Usage

  function Tail_After_Match (Source : String; Pattern : Character) return String;
  function Tail_After_Match (Source : String; Pattern : String) return String;

- Examples

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

Tio.Put_Line (Uxs.Tail_After_Match (Path, '/'));
"gnx-startup"

Put_Line (Uxs.Tail_After_Match (Path, "ix"));
"/gnx-startup"

Put_Line (Uxs.Tail_After_Match (Path, "gene"));
"six/gnx-startup"

Tio.Put_Line (Uxs.Tail_After_Match (Path, "etc/genesix/gnx-startu"));
"p"
```

```
Tio.Put_Line (Uxs.Tail_After_Match (Path, "/etc/genesix/gnx-startu"));
"p"

Tio.Put_Line (Uxs.Tail_After_Match (Path, "/etc/genesix/gnx-startup"));
empty string

Tio.Put_Line (Uxs.Tail_After_Match (Path, "/etc/genesix/gnx-startupp"));
empty string

Tio.Put_Line (Uxs.Tail_After_Match (Path, "/etc/geneseven"));
empty string
```

1.13.17 To_Hex

- Description

  Convert a Byte or String to a String hexadecimal output.

- Usage

  function To_Hex (Byte : Interfaces.Unsigned_8) return String
  function To_Hex (String_To_Convert : String) return String

- Example

```
tio.Put_Line (Uxs.To_Hex (+"ABCDEF"));

41 42 43 44 45 46
```

1.13.18 To_Hex_From_Val

- Description

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

- Usage

  function Ascii_Value_To_Hex (Input : String) return String

- Example

```
tio.Put_Line (Uxs.To_Hex_From_Val ("61"));

3D
```

1.13.19 To_Integer

- Description

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

- Usage

  function To_Integer (V : String) return Integer

- Example

```
tio.Put_Line (Uxs.To_Integer ("22"));
22
```

1.13.20 To_String

- Description

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

- Usage

  function To_String (B : Boolean) return String
  function To_String (I : Integer) return String
  function To_String (I : Long_Integer) return String;
  function To_String (C : ASCII_Character) return String

- Example

```
tio.Put_Line (Uxs.To_String (22));
"22"
```

1.13.21 To_Val

- Description

  Convert a String to String ASCII decimal values formatted output.

- Usage

  function To_Val (String_To_Convert : String) return String

- Example

```
Tio.Put_Line (Uxs.To_Val ("ABCDEF"));
65 66 67 68 69 70
```

1.13.22 Trim_Both

- Description

  Returns an all trimmed spaces String of String Source.

- Usage

  function Trim_Both (Source : String) return String

- Example

```
Tio.Put_Line (Uxs.Trim_Right ("   AB CD   "));
"AB CD"
```

1.13.23 Trim_Left

- Description

  Returns a trimmed leading spaces String of String Source.

- Usage

  function Trim_Left (Source : String) return String

- Example

```
Tio.Put_Line (Uxs.Trim_Left (+"   ABCD   "));
"ABCD   "
```

1.13.24 Trim_Right

- Description

  Returns a trimmed trailing spaces VString of VString Source.

- Usage

  function Trim_Right (Source : String) return String

- Example

```
Tio.Put_Line (Uxs.Trim_Right (+"   ABCD   "));
"   ABCD"
```

1.13.25 Trim_Slashes

- Description

  Returns an all trimmed slahes String of String Source.

- Usage

  function Trim_Slashes (Source : String) return String

- Example

```
Tio.Put_Line (Uxs.Trim_Slashes ("/"));
""

Tio.Put_Line (Uxs.Trim_Slashes ("I"));
"I"

Tio.Put_Line (Uxs.Trim_Slashes ("/i"));
"i"

Tio.Put_Line (Uxs.Trim_Slashes ("//////i/////"));
"i"
```

2 UXStrings

2.1 Types

```
type Encoding_Scheme is (ASCII_7, Latin_1, UTF_8, UTF_16BE, UTF_16LE);
-- Supported encoding schemes

subtype UTF_16_Encoding_Scheme is Encoding_Scheme range UTF_16BE .. UTF_16LE;
-- Supported UTF-16 encoding schemes

subtype ASCII_Character is Ada.Characters.Handling.ISO_646;
subtype ASCII_Character_Array is String with Dynamic_Predicate => (for all Item of
ASCII_Character_Array => Item in ASCII_Character);
-- Characters in ISO/IEC 646

subtype Latin_1_Character is Character;
subtype Latin_1_Character_Array is String;
-- Characters in ISO/IEC 8859-1

subtype BMP_Character is Wide_Character;
subtype BMP_Character_Array is Wide_String;
-- Characters in Unicode Basic Multilingual Plane
-- (Could be also named UCS_2_Character (Universal Coded Character Set)?)

subtype Unicode_Character is Wide_Wide_Character;
subtype Unicode_Character_Array is Wide_Wide_String;
-- Characters in Unicode planes
-- (Could be also named UCS_4_Character?)

subtype UTF_8_Character_Array is Ada.Strings.UTF_Encoding.UTF_String;
subtype UTF_16_Character_Array is Ada.Strings.UTF_Encoding.UTF_String;
-- Array of 8 bits values representing UTF encodings (UTF-8, UTF-16BE, or UTF-16LE)

type UXString is tagged private with
    Constant_Indexing => Element,
    Iterable => (First => First, Next => Next, Has_Element => Has_Element, Element =>
Element), String_Literal    => From_Unicode;
-- Container type of Unicode characters with dynamic size usually named string

Null_UXString : constant UXString;
-- Represent the null string
```

2.2 Functions and procedures

2.2.1 Append

- Description

  Update Source to the concatenation of Source and New_Item.

- Usage

```
procedure Append (Source : in out UXString; New_Item : UXString);
procedure Append (Source : in out UXString; New_Item : Unicode_Character);
```

- Example

2.2.2 Character_Set_Version

- Description

  Returns an implementation-defined identifier that identifies the version of the character set standard that is used for categorizing characters by the implementation.

- Usage

```
function Character_Set_Version return UXString;
```

- Example

2.2.3 Count

- Description

  Return the number of non overlapping occurrences of Pattern matching Source with respect of Mapping.

  Return the number of occurrences of characters in parameter Set matching Source

- Usage

```
function Count (Source : UXString; Pattern : UXString; Mapping : Wide_Wide_Character_Mapping := Identity) return Natural;

function Count (Source : UXString; Pattern : UXString; Mapping : Wide_Wide_Character_Mapping_Function) return Natural;

function Count (Source : UXString; Set : Wide_Wide_Character_Set) return Natural;
```

- Example

2.2.4 Delete

- Description

  Return Source whom characters with positions from Low to High are removed.

  Update Source whom characters with positions from Low to High are removed

- Usage

  function Delete (Source : UXString; From : Positive; Through : Natural) return UXString;

  procedure Delete (Source : in out UXString; From : Positive; Through : Natural);

- Example

2.2.5 Element

- Description

  Return the Unicode character of Source at Index position

- Usage

  function Element (Source : UXString; Index : Positive) return Unicode_Character;

- Example

2.2.6 Equal_Case_Insentive

- Description

  Returns True if the strings consist of the same sequence of characters after applying locale-independent simple case folding, as defined by documents referenced in the note in Clause 1 of ISO/IEC 10646:2011. Otherwise, returns False.

- Usage

  function Equal_Case_Insensitive (Left, Right : UXString) return Boolean;

- Example

2.2.7 Find_Token

- Description

  Set First to position of the first character inside or outside Set matches Source starting at From position. Set Last to position of the last character inside or outside Set matches Source with respect of Test membership

  Set First to position of the first character inside or outside Set matches Source. Set Last to position of the last character inside or outside Set matches Source with respect of Test membership.

- Usage

  procedure Find_Token (Source : UXString; Set : Wide_Wide_Character_Set; From : Positive; Test : Membership; First : out Positive; Last : out Natural);

  procedure Find_Token (Source : UXString; Set : Wide_Wide_Character_Set; Test : Membership; First : out Positive; Last : out Natural);

- Example

2.2.8 First

- Description

  Return the position of the first character of Source (actually 1).

- Usage

  function First (Source : UXString) return Positive;

- Example

2.2.9 From_ASCII

- Description

  Return an UXString from the ASCII character Item.
  Return an UXString from the array of ASCII characters Source.

- Usage

  function From_ASCII (Item : ASCII_Character) return UXString;
  function From_ASCII (Source : ASCII_Character_Array) return UXString;

- Example

2.2.10 From_BMP

- Description

  Return an UXString from the BMP character parameter Item.
  Return an UXString from the array of BMP characters parameter Source.

- Usage

  function From_BMP (Item : BMP_Character) return UXString;
  function From_BMP (Source : BMP_Character_Array) return UXString;

- Example

2.2.11 From_Latin_1

- Description

  Return an UXString from the Latin 1 character parameter Item.
  Return an UXString from the array of Latin 1 characters parameter Source.

- Usage

  function From_Latin_1 (Item : Latin_1_Character) return UXString;
  function From_Latin_1 (Source : Latin_1_Character_Array) return UXString;

- Example

2.2.12 From_Unicode

- Description

  Return an UXString from the Unicode character parameter Item.
  Return an UXString from the array of Unicode characters parameter Source.

- Usage

  function From_Unicode (Item : Unicode_Character) return UXString;

  function From_Unicode (Source : Unicode_Character_Array) return UXString;

- Example

2.2.13 From_UTF_8

- Description

  Return an UXString from the array of UTF-8 characters parameter Source, leading BOM characters are suppressed if any.

- Usage

  function From_UTF_8 (Source : UTF_8_Character_Array) return UXString;

- Example

2.2.14 From_UTF_16

- Description

  Return an UXString from the array of UTF-16 characters parameter Source according to the encoding scheme specified by Input_Scheme, leading BOM characters are suppressed if any.

- Usage

  function From_UTF_16 (Source : UTF_16_Character_Array; Input_Scheme : UTF_16_Encoding_Scheme) return UXString;

- Example

2.2.15 Get_ASCII

- Description

  Return the ASCII character of Source at Index position, if the character is not in ASCII set then Substitute is returned

- Usage

  Q_L : Latin_1_Character renames Ada.Characters.Latin_1.Question;

  function Get_ASCII (Source : UXString; Index : Positive; Substitute : in ASCII_Character := Q_L) return ASCII_Character;

- Example

2.2.16 Get_BMP

- Description

  Return the BMP character from Source at Index position, if the character is not in BMP set then Substitute is returned.

- Usage

  Inv_Q_B : BMP_Character renames Ada.Characters.Wide_Latin_1.Inverted_Question;

  function Get_BMP (Source : UXString; Index : Positive; Substitute : in BMP_Character := Inv_Q_B) return BMP_Character;

- Example

2.2.17 Get_Latin_1

- Description

  Return the Latin 1 character from Source at Index position, if the character is not in latin 1 set then Substitute is returned.

- Usage

  Inv_Q_L : Latin_1_Character renames Ada.Characters.Latin_1.Inverted_Question;

  function Get_Latin_1 (Source : UXString; Index : Positive; Substitute : in Latin_1_Character := Inv_Q_L) return Latin_1_Character;

- Example

2.2.18 Get_Unicode

- Description

  Return the Unicode character from Source at Index position.

- Usage

  function Get_Unicode (Source : UXString; Index : Positive) return Unicode_Character;

- Example

2.2.19 Has_Element

- Description

  Return True if a character of Source is present at Index position (actually Index <= Length (Source)).

- Usage

  function Has_Element (Source : UXString; Index : Positive) return Boolean;

- Example

2.2.20 Head

- Description

  Return the first characters from Source up to Count concatenated with Pad characters if needed

  Update Source to the first characters from Source up to Count concatenated with Pad characters if needed

- Usage

  function Head (Source : UXString; Count : Natural; Pad : Unicode_Character := Wide_Wide_Space) return UXString;

  procedure Head (Source : in out UXString; Count : Natural; Pad : Unicode_Character := Wide_Wide_Space);

- Example

2.2.21 Index

- Description

  (1) Return the position of the first character where Pattern matches Source with respect of Going direction and Mapping

  (2) Return the position of the first character where Pattern matches Source with respect of Going direction and Mapping

[3] Return the position of the first character inside or outside Set matches Source with respect of Going direction and Test membership

[4] Return the position of the first character where Pattern matches Source starting at From position with respect of Going direction and Mapping

[5] Return the position of the first character where Pattern matches Source starting at From position with respect of Going direction and Mapping

[6] Return the position of the first character inside or outside Set matches Source starting at From position with respect of Test membership

- Usage

[1] function Index (Source : UXString; Pattern : UXString; Going : Direction := Forward; Mapping : Wide_Wide_Character_Mapping := Identity) return Natural;

[2] function Index (Source : UXString; Pattern : UXString; Going : Direction := Forward; Mapping : Wide_Wide_Character_Mapping_Function) return Natural;

[3] function Index (Source : UXString; Set : Wide_Wide_Character_Set; Test : Membership := Inside; Going : Direction := Forward) return Natural;

[4] function Index (Source : UXString; Pattern : UXString; From : Positive; Going : Direction := Forward; Mapping : Wide_Wide_Character_Mapping := Identity) return Natural;

[5] function Index (Source : UXString; Pattern : UXString; From : Positive; Going : Direction := Forward; Mapping : Wide_Wide_Character_Mapping_Function) return Natural;

[6] function Index (Source : UXString; Set : Wide_Wide_Character_Set; From : Positive; Test : Membership := Inside; Going : Direction := Forward) return Natural;

```
With Ada.Strings - Direction is (Forward, Backward);
   type Alignment  is (Left, Right, Center);
   type Truncation is (Left, Right, Error);
   type Membership is (Inside, Outside);

   type Trim_End   is (Left, Right, Both);
```

- Example

2.2.22 Index_Non_Blank

- Description

Return the position of the first non space character of Source with respect of Going direction.

Return the position of the first non space character of Source starting at From position with respect of Going direction.

- Usage

```
function Index_Non_Blank (Source : UXString; Going : Direction := Forward) return Natural;
function Index_Non_Blank (Source : UXString; From : Positive; Going : Direction := Forward) return Natural;
```

- Example

```

```

2.2.23 Insert

- Description

Return Source with New_Item inserted at position ahead of parameter Before.
Update Source with New_Item inserted at position ahead of parameter Before.

- Usage

```
function Insert (Source : UXString; Before : Positive; New_Item : UXString) return UXString;
procedure Insert (Source : in out UXString; Before : Positive; New_Item : UXString);
```

- Example

```

```

2.2.24 Is_ASCII, Is_ISO_646

- Description

Return True if all the characters of Source are in ASCII set.
Return True if the character of Source at Index position is in ASCII set.

- Usage

```
function Is_ASCII (Source : UXString) return Boolean;
function Is_ASCII (Source : UXString; Index : Positive) return Boolean;

function Is_ISO_646 (Item : UXString) return Boolean renames Is_ASCII;
```

- Example

```

```

2.2.25 Is_BMP

- Description

  Return True if all the characters of Source are in BMP set.
  Return True if the character of Source at Index position is in BMP set

- Usage

```
function Is_BMP (Source : UXString) return Boolean;
function Is_BMP (Source : UXString; Index : Positive) return Boolean;
```

- Example

```
```

2.2.26 Is_Latin_1

- Description

  Return True if all the characters of Source are in Latin 1 set.
  Return True if the character of Source at Index position is in Latin 1 set.

- Usage

```
Inv_Q_L : Latin_1_Character renames Ada.Characters.Latin_1.Inverted_Question;

function Is_Latin_1 (Source : UXString) return Boolean;
function Is_Latin_1 (Source : UXString; Index : Positive) return Boolean;
```

- Example

```
```

2.2.27 Is_Unicode

- Description

  Return True if all the characters of Source are in Unicode set (actually True).
  Return True if the character of Source at Index position is in Unicode set (actually True).

- Usage

```
function Is_Unicode (Source : UXString) return Boolean;
function Is_Unicode (Source : UXString; Index : Positive) return Boolean;
```

- Example

```
```

2.2.28 Last

- Description

  Return the position of the last character of Source (actually Length (Source)).

- Usage

  function Last (Source : UXString) return Natural;

- Example

2.2.29 Length

- Description

  Return the number of (Unicode) characters.

- Usage

  function Length (Source : UXString) return Natural;

- Example

2.2.30 Less_Case_Insensitive

- Description

  Performs a lexicographic comparison of strings Left and Right, converted to lower case.

- Usage

  function Less_Case_Insensitive (Left, Right : UXString) return Boolean;

- Example

2.2.31 Next

- Description

  Return the position of the next character of Source after Index position (actually Index + 1).

- Usage

  function Next (Source : UXString; Index : Positive) return Positive;
  procedure Next (Source : UXString; Index : in out Positive);

- Example

2.2.32 Overwrite

- Description

  Return Source whom characters starting at Position are replaced with parameter New_Item
  Update Source whom characters starting at Position are replaced with parameter New_Item

- Usage

  function Overwrite (Source : UXString; Position : Positive; New_Item : UXString) return UXString;
  procedure Overwrite (Source : in out UXString; Position : Positive; New_Item : UXString);

- Example

2.2.33 Prepend

- Description

  Update Source to the concatenation of New_Item and Source

- Usage

  procedure Prepend (Source : in out UXString; New_Item : UXString);
  procedure Prepend (Source : in out UXString; New_Item : Unicode_Character);

- Example

2.2.34 Replace_ASCII

- Description

  Update Source such as the character at Index position is set to the ASCII character parameter By.

- Usage

  procedure Replace_ASCII (Source : in out UXString; Index : Positive; By : ASCII_Character);

- Example

2.2.35 Replace_Latin_1

- Description

  Update Source such as the character at Index position is set to the Latin 1 character parameter By.

- Usage

  procedure Replace_Latin_1 (Source : in out UXString; Index : Positive; By : Latin_1_Character);

- Example

2.2.36 Replace_BMP

- Description

  Update Source such as the character at Index position is set to the BMP character parameter By.

- Usage

  procedure Replace_BMP (Source : in out UXString; Index : Positive; By : BMP_Character);

- Example

2.2.37 Replace_Slice

- Description

  Return Source whom characters with positions from Low to High are replaced with parameter By.

  Update Source whom characters with positions from Low to High are replaced with parameter By.

- Usage

  function Replace_Slice (Source : UXString; Low : Positive; High : Natural; By : UXString) return UXString;

  procedure Replace_Slice (Source : in out UXString; Low : Positive; High : Natural; By : UXString);

- Example

2.2.38 Replace_Unicode

- Description

  Update Source such as the character at Index position is set to the Unicode character parameter By.

- Usage

  procedure Replace_Unicode (Source : in out UXString; Index : Positive; By : Unicode_Character);

- Example

2.2.39 Set

- Description

  Set Target to an UXString from the array of Unicode characters parameter Source.

- Usage

  procedure Set (Target : out UXString; Source : Unicode_Character_Array);

- Example

2.2.40 Slice

- Description

  Return the slice at positions Low through High from Source.

  Set Target to the slice at positions Low through High from Source.

- Usage

  function Slice (Source : UXString; Low : Positive; High : Natural) return UXString;

  procedure Slice (Source : UXString; Target : out UXString; Low : Positive; High : Natural);

- Example

2.2.41 Tail

- Description

  Return the last characters from Source up to Count concatenated with Pad characters if needed

  Update Source to the last characters from Source up to Count concatenated with Pad characters if needed

- Usage

  function Tail (Source : UXString; Count : Natural; Pad : Unicode_Character := Wide_Wide_Space) return UXString;

  procedure Tail (Source : in out UXString; Count : Natural; Pad : Unicode_Character := Wide_Wide_Space);

- Example

2.2.42 To_ASCII, To_ISO_646

- Description

  Return an array of ASCII characters from Source, if a character is not in ASCII set then Substitute is returned.

- Usage

  Q_L : Latin_1_Character renames Ada.Characters.Latin_1.Question;

  function To_ASCII (Source : UXString; Substitute : in ASCII_Character := Q_L) return ASCII_Character_Array;

  function To_ISO_646 (Item : UXString; Substitute : Ada.Characters.Handling.ISO_646 := Q_L) return ASCII_Character_Array renames To_ASCII;

- Example

2.2.43 To_Basic

- Description

  Returns the letter corresponding to Item but with no diacritical mark, if Item is a letter but not a basic letter; returns Item otherwise.

- Usage

  function To_Basic (Item : UXString) return UXString;

- Example

2.2.44 To_BMP

- Description

  Return an array of BMP characters from Source, if a character is not in BMP set then Substitute is returned.

- Usage

  Inv_Q_B : BMP_Character renames Ada.Characters.Wide_Latin_1.Inverted_Question;

  function To_BMP (Source : UXString; Substitute : in BMP_Character := Inv_Q_B) return BMP_Character_Array;

- Example

2.2.45 To_Latin_1

- Description

  Return an array of Latin 1 characters from Source, if a character is not in latin 1 set then Substitute is returned.

- Usage

  Inv_Q_L : Latin_1_Character renames Ada.Characters.Latin_1.Inverted_Question;

  function To_Latin_1 (Source : UXString; Substitute : in Latin_1_Character := Inv_Q_L) return Latin_1_Character_Array;

- Example

2.2.46 To_Lower

- Description

  Returns the corresponding lower-case value for Item if Is_Upper(Item), and returns Item otherwise.

- Usage

  function To_Lower (Item : UXString) return UXString;

- Example

2.2.47 To_Unicode

- Description

  Return an array of Unicode characters from Source.

- Usage

  function To_Unicode (Source : UXString) return Unicode_Character_Array;

- Example

2.2.48 To_Upper

- Description

  Returns the corresponding upper-case value for Item if Is_Lower(Item) and Item has an upper-case form, and returns Item otherwise.

- Usage

  function To_Upper (Item : UXString) return UXString;

- Example

2.2.49 To_UTF_8

- Description

  Return an array of UTF-8 characters from Source, prepend UTF-8 BOM if Output_BOM is set to True.

- Usage

  function To_UTF_8 (Source : UXString; Output_BOM : Boolean := False) return UTF_8_Character_Array;

- Example

2.2.50 To_UTF_16

- Description

  Return an array of UTF-16 characters from Source according to the encoding scheme specified by Output_Scheme, prepend UTF-16 BOM if Output_BOM is set to True.

- Usage

  function To_UTF_16 (Source : UXString; Output_Scheme : UTF_16_Encoding_Scheme; Output_BOM : Boolean := False) return UTF_16_Character_Array;

- Example

2.2.51 Translate

- Description

  Return Source updated with respect of Mapping.

  Update Source with respect of Mapping.

  Return Source updated with respect of Mapping

  Update Source with respect of Mapping

- Usage

  function Translate (Source : UXString; Mapping : Wide_Wide_Character_Mapping) return UXString;

  procedure Translate (Source : in out UXString; Mapping : Wide_Wide_Character_Mapping);

  function Translate (Source : UXString; Mapping: Wide_Wide_Character_Mapping_Function) return UXString;
  procedure Translate (Source : in out UXString; Mapping : Wide_Wide_Character_Mapping_Function);

- Example

2.2.52 Trim

- Description

  Return Source with Space characters removed from left, right or both with respect of Side.

  Update Source with Space characters removed from left, right or both with respect of Side.

  Return Source with leading characters in Left and trailing characters in Right removed

  Update Source with leading characters in Left and trailing characters in Right removed

- Usage

  function Trim (Source : UXString; Side : Trim_End) return UXString;

  procedure Trim (Source : in out UXString; Side : Trim_End);

  function Trim (Source : UXString; Left : Wide_Wide_Character_Set; Right : Wide_Wide_Character_Set) return UXString;

  procedure Trim (Source : in out UXString; Left : Wide_Wide_Character_Set; Right : Wide_Wide_Character_Set);

  With Ada.Strings - type Trim_End is (Left, Right, Both);

- Example

2.2.53 &

- Description

  Return the concatenation of Left and Right.

- Usage

  function "&" (Left : UXString; Right : UXString) return UXString;
  function "&" (Left : UXString; Right : Unicode_Character) return UXString;
  function "&" (Left : Unicode_Character; Right : UXString) return UXString;

- Example

2.2.54 *

- Description

  Return Right string duplicated Left times

  Return Right character duplicated Left times

- Usage

  function "*" (Left : Natural; Right : UXString) return UXString;

  function "*" (Left : Natural; Right : Unicode_Character) return UXString;

- Example

2.2.55 =

- Description

  Return True if Left equals Right.

- Usage

  function "=" (Left : UXString; Right : UXString) return Boolean;;

- Example

2.2.56 <

- Description

  Return True if Left is less than Right.

- Usage

  function "<" (Left : UXString; Right : UXString) return Boolean;

- Example

2.2.57 <=

- Description

  Return True if Left is less or equal than Right.

- Usage

  function "<=" (Left : UXString; Right : UXString) return Boolean;

- Example

2.2.58 >

- Description

  Return True if Left is greater than Right.

- Usage

  function ">" (Left : UXString; Right : UXString) return Boolean;

- Example

2.2.59 >=

- Description

  Return True if Left greater or equal than Right.

- Usage

  function ">=" (Left : UXString; Right : UXString) return Boolean;

- Example

3 Gnoga

The Gnoga API is too large to be fully reproduced here. Please refer to the automatically generated HTML documentation available from GnatStudio menu.

3.1 Server.Database

3.1.1 Types

```
type Connection is limited interface;
type Connection_Access is access all Connection'Class;
```

```
type Recordset is interface;
```

```
type Recordset_Access is access all Recordset'Class;
```

```
type Field_Description is record
      Column_Name   : String;
      Data_Type     : String;
      Can_Be_Null   : Boolean;
      Default_Value : String;
end record;

package Field_Description_Arrays is new Ada.Containers.Indefinite_Vectors (Natural,
Field_Description);
subtype Field_Description_Array_Type is Field_Description_Arrays.Vector;
```

3.1.2 Exceptions

```
Connection_Error : Unable to connect to MYSQL Server or Not connected to Server

Database_Error : Unable to switch to specified Database

Table_Error : Unable to locate table or table has no fields

Query_Error : Unable to execute query

Empty_Recordset_Error : The recordset is currently empty

Empty_Row_Error : Attempt to read value from Row before calling Next

End_Of_Recordset : Attempt to go pass the last row in recordset

No_Such_Field : The value for a field name was requested that does not exits

Null_Field : The value for a Null field was requested

Not_Implemented : If a database method is called that is not implemented by the spe-
cific database engined used this exception will be raised.
```

3.1.3 Affected_Rows

- Description

  Returns the number of rows affected by an Execute_Query.

- Usage

  function Affected_Rows (C : Connection) return Natural

- Example

```

```

3.1.4 Close

- Description

  Close current recordset and free resources.

- Usage

  procedure Close (RS : in out Recordset)

- Example

```

```

3.1.5 Disconnect

- Description

  Disconnect from server.

- Usage

  procedure Disconnect (C : in out Connection)

- Example

```

```

3.1.6 Error_Message

- Description

  Returns the last error message that has occurred on this connection.

- Usage

  function Error_Message (C : Connection) return String

- Example

```

```

3.1.7 Escape_String

- Description

  Prepares a string for safe storage in a query.

- Usage

  function Escape_String (C : Connection; S : String) return String

- Example

```

```

3.1.8 Execute_Query

- Description

  Execute an SQL Query with no result set.

- Usage

  procedure Execute_Query (C : in out Connection; SQL : String)

- Example

3.1.9 Execute_Update

- Description

  Executes and SQL Query and returns the number of affected row.

- Usage

  function Execute_Update (C : in out Connection; SQL : String) return Natural

- Example

3.1.10 Field_Decimals

- Description

  Returns the decimal portion of a field size if it exists or 0. For example: if the Data_Type = float(10,2) it will return 2.

- Usage

  function Field_Decimals (Field : Field_Description) return Natural

- Example

3.1.11 Field_Descriptions

- Description

  Return an array of Field_Description records describe the fields of a table.

- Usage

  function Field_Descriptions (C : Connection; Table_Name : String) return Field_Description_Array_Type

- Example

3.1.12 Field_Name

- Description

  Return name of field.

- Usage

  function Field_Name (RS : Recordset; Field_Number : Natural) return String

- Example

3.1.13 Field_Options

- Description

  Returns the field options portion of a data type, for example: If the Field.Data_Type = enum('N','Y') then will return 'N','Y' as this is described in the database in the same way as field size, this may be used for a string representation of the size as well. For example varchar(80) will return the string 80. This is also used for descriptions like decimal(10,2), etc.

- Usage

  function Field_Options (Field : Field_Description) return String

- Example

3.1.14 Field_Options

- Description

  Returns the field size portion of a data type, for example: If the Field.Data_Type = varchar(80) then will return 80. If the Data_Type does not have a size portion will return 0. If the Data_Type is a numeric with decimals, e.g. decimal(10,2) then it will return the non-decimal portion.

- Usage

  function Field_Size (Field : Field_Description) return Natural

- Example

3.1.15 Field_Type

- Description

  Returns the field type portion of a data type, for example: If the Field.Data_Type = Varchar(80) then will return Varchar.

- Usage

  function Field_Type (Field : Field_Description) return String

- Example

3.1.16 Field_Value

- Description

  Return value of field, if Handle_Nulls is true, Null values will return as empty Strings.

- Usage

  function Field_Value (RS : Recordset; Field_Number : Natural; Handle_Nulls : Boolean := True) return String
  function Field_Value (RS : Recordset; Field_Name : String; Handle_Nulls : Boolean := True) return String

- Example

3.1.17 Field_Values

- Description

  Return map of all values for current row, NULL values are set to an empty String.

- Usage

  function Field_Values (RS : Recordset) return Gnoga.Types.Data_Map_Type

- Example

3.1.18 ID_Field_String

- Description

  Returns the proper type format for the ID field.
  For MySQL = "id INTEGER PRIMARY KEY AUTO_INCREMENT".
  For SQLite = "id INTEGER PRIMARY KEY AUTOINCREMENT".

- Usage

  function ID_Field_String (C : Connection) return String

- Example

3.1.19 Insert_ID

- Description

  Returns the last value assigned to an auto increment field upon insert.

- Usage

  function Insert_ID (C : Connection) return Natural

- Example

3.1.20 Is_Null

- Description

  Return True if value of field is null.

- Usage

  function Is_Null (RS : Recordset; Field_Number : Natural) return Boolean
  function Is_Null (RS : Recordset; Field_Name : String) return Boolean

- Example

3.1.21 Iterate

- Description

  [1] Iterate through all rows in the result set of the query
  [2] Iterate through all rows in the recordset
  [3] Iterate through all rows in the result set of the query
  [4] Iterate through all rows in the recordset

- Usage

  [1] procedure Iterate (C : in out Connection; SQL : in String; Process : not null access procedure (RS : Recordset'Class))

  [2] procedure Iterate (RS : in out Recordset; Process : not null access procedure (RS : Recordset'Class))

  [3] procedure Iterate (C : in out Connection; SQL : String; Process : not null access procedure (Row : Gnoga.Types.Data_Map_Type))

  [4] procedure Iterate (RS : in out Recordset; Process : not null access procedure (Row : Gnoga.Types.Data_Map_Type))

- Example

3.1.22 List_Of_Tables

- Description

  Return an array of table names

- Usage

  function List_Of_Tables (C : Connection) return Gnoga.Types.Data_Array_Type

- Example

3.1.23 List_Fields_Of_Table

- Description

  Return an array of field names for table.

- Usage

  function List_Fields_Of_Table (C : Connection; Table_Name : String) return Gnoga.Types.Data_Array_Type

- Example

3.1.24 Next

- Description

  Go to next row or Go to next row and return true if not End of Recordset.

- Usage

  procedure Next (RS : in out Recordset)
  function Next (RS : in out Recordset) return Boolean

- Example

3.1.25 Number_Of_Fields

- Description

  Return number of fields (columns) in recordset.

- Usage

  function Number_Of_Fields (RS : Recordset) return Natural

- Example

3.1.26 Query

- Description

  Execute query that returns Recordset.

- Usage

  function Query (C : Connection; SQL : String) return Recordset'Class

- Example

3.2 Application, Types, Gui, Server, Client

The Gnoga framework's root package is Gnoga. There are five child packages making up the five areas of Gnoga development.

Gnoga.Application and its children are related to initializing and managing the life cycle of Gnoga applications.

Gnoga.Types contains Gnoga specific types used throughout the framework.

Gnoga.Gui contains the user-interface portions of Gnoga. It is further divided into:
- Gnoga.Gui.Base - Common base functionality and events to all UI objects
- Gnoga.Gui.Document - Binding to root element of DOM in a window
- Gnoga.Gui.Element - General binding to all UI objects
- Gnoga.Gui.Element.Common - Common UI elements
- Gnoga.Gui.Element.Form - Form-related UI elements
- Gnoga.Gui.Ekement.Canvas - Binding to a drawing canvas
- Gnoga.Gui.Element.Multimedia - Multimedia bindings
- Gnoga.Gui.Element.SVG - SVG canvas binding
- Gnoga.Gui.Location - Browser window location control
- Gnoga.Gui.Navigator - Browser application control
- Gnoga.Gui.Screen - Desktop screen properties
- Gnoga.Gui.View - Layout control of UI elements
- Gnoga.Gui.Window - Control of connection to UI

Gnoga.Server - Server side bindings and features
- Gnoga.Server - Application settings and directories
- Gnoga.Server.Connection - Low level control of connection to UI
- Gnoga.Server.Database - Database bindings [MySQL and SQLite3]
- Gnoga.Server.Migration - Database schema migration interface
- Gnoga.Server.Model - Active Record implementation for Database access
- Gnoga.Server.Template_Parser - Template parsing [Python or simple text]

Gnoga.Client - Non GUI client side bindings
- Gnoga.Client.Storage - Local storage on client side
- Gnoga.Client.Bind_Page - Dynamically create Gnoga objects for an HTML page

3.3 Hierarchy for GUI Types

```
Gnoga.GUI.Base.Base_Type
|
|__ Gnoga.Gui.Document.Document_Type
|
|__ Gnoga.Gui.Element.Element_Type
    |
    |__ Gnoga.Gui.Element.Canvas.Canvas_Type
    |                       Context_Type
    |                       |
    |                       |__ Canvas.Context_2d.Context_2d_Type
    |
    |__ Gnoga.Gui.Element.Common.A_Type
    |                             Button_Type
    |                             Div_Type
    |                             P_Type
    |                             IMG_Type
    |                             HR_Type
    |                             BR_Type
    |                             Meter_Type
    |                             Progress_Bar_Type
    |                             Span_Type
    |
    |__ Gnoga.Gui.Element.Form.Form_Type
    |                         |    Form_Element_Type
    |                         |    Data_List_Type
    |                         |    Text_Area_Type
    |                         |    Hidden_Type
    |                         |    Input_Button_Type
    |                         |    Submit_Button_Type
    |                         |    Reset_Button_Type
    |                         |    Check_Box_Type
    |                         |    Radio_Button_Type
    |                         |    Check_Box_Type
```

```
|                        |    Input_Image_Type
|                        |    Text_Type
|                        |    Email_Type
|                        |    Password_Type
|                        |    URL_Type
|                        |    Search_Type
|                        |    Color_Picker_Type
|                        |    Date_Type
|                        |    Time_Type
|                        |    Month_Type
|                        |    Week_Type
|                        |    Date_Time_Type
|                        |    Date_Time_Local_Type
|                        |    Number_Type
|                        |    Range_Type
|                        |    Label_Type
|                        |    Selection_Type
|                        |    Option_Type
|                        |    Option_Group
|                        |__ Form.Fieldset.Fieldset_Type
|
|__ Gnoga.Gui.Element.IFrame.IFrame_Type
|
|__ Gnoga.Gui.Element.List.Ordered_List_Type
|                           Unordered_List_Type
|                           List_Item_Type
|                           Definition_List_Type
|                           Term_Type
|                           Description_Type
|
|__ Gnoga.Gui.Element.Multimedia.Multimedia_Type
|                                 Audio_Type
|                                 Video_Type
|
|__ Gnoga.Gui.Element.Phrase.Phase_Type (Abbr, Code, Strong, Em, Dfn, Samp,
|                                        Kbd, Var, Marked, Del, Ins, S, Q,
|                                        Big, Small, Time, Tt, Cite, I, B,
|                                        U, Sub, Sup)
|
|__ Gnoga.Gui.Element.Section.Section_Type (Address, Article, Aside, Header,
|                                           Main, Nav, P, Pre, Section,
|                                           BlockQuote, H1, H2, H3, H4, H5,
|                                           H6, HGroup)
|
|__ Gnoga.Gui.Element.Style.Style_Block.Style_Type
|
|__ Gnoga.Gui.Element.SVG.SVG_Type
|
|__ Gnoga.Gui.Element.Table.Table_Type
|                           Table_Row_Type
|                           Table_Column_Type
|                           Table_Heading_Type
|                           Table_Header_Type
|                           Table_Body_Type
|                           Table_Footer_Type
|                           Table_Group_Type
|                           Table_Column_Type
|
|__ Gnoga.Gui.Location.Location_Type
|
|__ Gnoga.Gui.Module - Place holder for 3rd party extensions
|
|__ Gnoga.Gui.Navigator
|
|__ Gnoga.Gui.Plugin - Place hordler for 3rd part JS bindings
|          included - Ace_Editor
|                      Boot_Strap
|                      |
|                      |__ Container_Type
|                          Fluid_Container_Type
```


```
|                            Row_Type
|                            Jumbotron_Type
|                            Table_Type
|                            Form_Group_Type
|                            Check_Box_Type
|                            Radio_Button_Type
|                         jQuery - Additional to Gnoga's use
|                         jQueryUI - Interactions, Effects, Utilities
|                         |
|                         |__ jQueryUI.Widget.Accordian_Type
|                                              jQueryUI Button
|                                              Dialog_Type
|                                              Progress_Bar_Type
|                                              jQueryUI Menu
|                                              jQueryUI Select Menu
|                                              Tabs_Type
|                                              jQueryUI Tool Tips
|                         MacGap - Native Mac OS X functionality
|__ Gnoga.Gui.Screen
|
|__ Gnoga.Gui.View.View_Base_Type
|              |    View_Type
|              |
|              |__ View.Card.Card_View_Type
|              |             Tab_Type
|              |             Tab_Item_Type
|              |
|              |__ View.Console.Console_View_Type
|              |
|              |__ View.Docker.Docker_View_Type
|
|__ Gnoga.Gui.Window.Window_Type
```

3.4 Property and Method Overview

3.4.1 Base_Type

```
* Properties
   - Height
   - Width

* Frawmework Properties:
   - Buffer_Connection
   - Connection_Data [ro]
   - Connection_ID
   - DOM_Selector [ro]
   - Dynamic
   - ID
   - ID_Type [ro]
   - Parent
   - Unique_ID [ro]
   - Valid [ro]

* Generic Client Side Access to Properties
   - Property

* Methods
   - Focus
   - Blur

* Framework Methods
   - Flush_Buffer

* Generic Client Side Execution of Methods
   - Execute
```

3.4.2 Element_Type

All of Base_Type Plus:

```
   * Properties - General
      - Access_Key
      - Advisory_Title
      - Class_Name
      - Draggable
      - Editable
      - Inner_HTML
      - Outer_HTML [ro]

   * Properties - Text Content
      - Language_Code
      - Tab_Index
      - Spell_Check
      - Text
      - Text_Direction

   * Properties - Visibility and Layout
      - Hidden
      - Visible
      - Display
      - Clear_Side [wo]
      - Layout_Float [wo]
      - Overflow
      - Overflow_X [wo]
      - Overflow_Y [wo]
      - Resizable
      - Position
      - Verticle_Align [wo]
      - Box_Sizing
      - Z_Index [wo]
      - Margin [wo]
      - Padding [wo]

   * Properties - Position
      - Height [from Base_Type]
      - Width [from Base_Type]
      - Position_Top [ro]
      - Position_Left [ro]
      - Offset_From_Top [ro]
      - Offset_From_Left [ro]
      - Left
      - Right
      - Top
      - Bottom
      - Box_Height
      - Box_Width
      - Minimum_Height
      - Minimum_Width
      - Maximum_Height
      - Maximum_Width
      - Inner_Height
      - Inner_Width
      - Outer_Height [ro]
      - Outer_Width [ro]
      - Outer_Height_To_Margin [ro]
      - Outer_Width_To_Margin [ro]
      - Client_Width [ro]
      - Client_Height [ro]
      - Client_Left [ro]
      - Client_Top [ro]
      - Offset_Width [ro]
      - Offset_Height [ro]
      - Offset_Left [ro]
      - Offset_Top [ro]
```

```
      - Scroll_Width [ro]
      - Scroll_Height [ro]
      - Scroll_Left
      - Scroll_Top

   * Properties - Style - Color
      - Color
      - Opacity
      - Background_Attachment
      - Background_Color
      - Background_Image
      - Background_Position
      - Background_Origin
      - Background_Repeat
      - Background_Clip
      - Background_Size
      - Border [wo]
      - Border_Radius [wo]
      - Shadow [wo] / Shadow_None
      - Outline [wo]
      - Cursor

   * Properties - Style - Text
      - Font [wo]
      - Text_Alignment [wo]

   * Framework Properties:
      - Auto_Place
      - First_Child
      - Next_Sibling
      - HTML_Tag [ro]

   * Generic Client Side Access to Properties
      - Style
      - Attribute

   * Methods
      - Click
      - Add_Class
      - Remove_Class
      - Toggle_Class
      - Place_Inside_Top_Of
      - Place_Inside_Bottom_Of
      - Place_Before
      - Place_After
      - Remove
```

3.4.3 Events

```
   * Object Events
      - On_Resize
      - On_Scroll

   * Form Events
      - On_Focus
      - On_Blur
      - On_Change
      - On_Focus_In
      - On_Focus_Out
      - On_Input
      - On_Reset
      - On_Search
      - On_Select
      - On_Submit

   * Mouse Events
      - On_Click
```

```
      - On_Mouse_Click
      - On_Mouse_Right_Click
      - On_Context_Menu
      - On_Double_Click
      - On_Mouse_Double_Click
      - On_Mouse_Enter
      - On_Mouse_Leave
      - On_Mouse_Over
      - On_Mouse_Out
      - On_Mouse_Down
      - On_Mouse_Up
      - On_Mouse_Move

   * Drag and Drop Events
      - On_Drag_Start
      - On_Drag
      - On_Drag_End
      - On_Drag_Enter
      - On_Drag_Leave
      - On_Drop

   * Keyboard Events
      - On_Character
      - On_Wide_Character
      - On_Key_Down
      - On_Key_Up
      - On_Key_Press

    * Clipboard Events
      - On_Copy
      - On_Cut
      - On_Paste

    * Generic Events
      - On_Create
      - On_Destroy
      - On_Child_Added
      - On_Child_Removed
      - On_Message
```

FAQ

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

1 Constants

1.1 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";
```

1.2 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;
```

1.3 Delimiter characters

V22 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.

1.4 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";
```

1.5 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";
```

2 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 v22. 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 :

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

3 Converting reminder

3.1 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”
```

3.2 Converting a character to its ASCII value

65 is ASCII code for ‘A’ :

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

3.3 Converting String from and to Long_Integer

```
procedure Test is

      Test_String : String := "";
      Test_Long_Integer : Long_Integer := 10737418240;

   begin

      Test_String := To_String [Test_Long_Integer]; -- Test_Long_Integer´Image

      Msg.Std [Test_String];

      Msg.Std [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;
```

Implementation

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 Introduction

Implementation notes subject to change. To be seen for now as a working document.

2 Ada

2.1 Ada.Containers.Vectors with records

2.1.1 Declarations

2.1.2 Write

2.1.3 Read

```
DB_Index := Databases.First_Index;
Msg.Dbg ("Test DB_Properties: " & Databases(DB_Index).Name);
```

2.1.4 Iterates

```
   DB_Cursor : Databases_List.Cursor := Databases_List.No_Element;
begin
   while DB_Cursor /= Databases_List.No_Element loop
      DB_Cursor := Next (DB_Cursor);
   end loop;
   DB_Index := To_Index (DB_Cursor);

   for I of Databases loop
      if I.Name = DB_Name then
         exit;
      end if;
   end loop;
```

2.1.5 Search

```
- Test
  DBI := Sql.DB_Index ("v23");
  if DBI /= Sql.Databases_List.No_Index then
     Msg.Dbg ("DB_Index found: " & From_Latin_1 (DBI'Image));
     Msg.Dbg ("URI from Index: " & Sql.Databases(DBI).URI);
  else
     Msg.Err ("DB_Index not found: v23");
  end if;

- DB_Index not implemented in the API, as it required the Databases container instance
to be public

function DB_Index (DB_Name : String) return Databases_List.Extended_Index is
   use Databases_List; -- for operators
   DB_Index : Databases_List.Extended_Index := Databases_List.No_Index;
begin

   for C in Databases.Iterate loop
      --Msg.Dbg ("Index: " & From_Latin_1 (Databases_List.Extended_Index'Image (To_In-
dex (C))));
   if  Databases(C).Name = DB_Name then
        DB_Index := Databases_List.Extended_Index (To_Index (C));
      exit;
      end if;
   end loop;
   return DB_Index;
end DB_Index;

function DB_Properties (DB_Name : String) return Database_Line is
   use Databases_List; -- for operators
   DB_Index : Databases_List.Extended_Index := Databases_List.No_Index;
   DB_Record : Database_Line;
begin
   for C in Databases.Iterate loop
      --Msg.Dbg ("Index: " & From_Latin_1 (Databases_List.Extended_Index'Image (To_In-
dex (C))));
   if  Databases(C).Name = DB_Name then
        DB_Record := Databases(C);
      exit;
      end if;
   end loop;
  return DB_Record;
end DB_Properties;
```

3 API Rest

geta : aws ada api rest

max url length : 2K (some browsers) or 8K per convention

https://en.wikibooks.org/wiki/Ada_Programming/Libraries/Web/AWS

https://github.com/BrentSeidel/BBS-BBB-Ada/tree/master
https://github.com/BrentSeidel/BBS-Ada

https://docs.adacore.com/aws-docs/aws

https://restfulapi.net/rest-put-vs-post
https://www.baeldung.com/rest-http-put-vs-post

https://blog.apilayer.com/an-ultimate-guide-to-http-put-vs-post-in-rest-api-in-2023

Use get & post [see API IONOS]

4 CRC

https://crccalc.com [outil interactif 8/16/32 bits]

https://sourceforge.net/p/avr-ada/wiki/CRC

4.1 CRC 16

https://srecord.sourceforge.net/crc16-ccitt.html

http://computer-programming-forum.com/44-ada/ce7afed4795fb0e4.htm
https://www.avrfreaks.net/s/topic/a5C3l000000UaabEAC/t154443
https://www.nongnu.org/avr-libc/user-manual/group__util__crc.html
https://www.carnetdumaker.net/snippets/29

4.2 CRC 32

https://rosettacode.org/wiki/CRC-32

5 Databases

5.1 Introduction

<<<<TODO>>>

5.2 Benchmarking

5.2.1 Locust

https://simonwillison.net/2022/Oct/23/datasette-gunicorn/#benchmarking-sqlite

6 Database MySQL

6.1 Transactions sequences

```
DB.Execute_Query ("START TRANSACTION");
DB.Execute_Query ("SAVEPOINT SV_Index_Exists");
 <<<work>>>
DB.Execute_Query ("COMMIT");
```

<<<TODO>>>

7 Database SQLite

7.1 Transactions sequences

```
DB.Execute_Query ("BEGIN TRANSACTION");
DB.Execute_Query ("SAVEPOINT SV_Index_Exists");
 <<<work>>>
DB.Execute_Query ("COMMIT");
```

<<<TODO>>>

7.2 Compatibility with MySQL

7.2.1 Problem description

Number_Of_Rows is not implemented in SQLite: this is acceptable because it is clearly expressed in the doc and it can be implemented with the reservations below.

Affected_Rows is not implemented correctly for SQLite: this is annoying because it's not expressed at all as Gnoga doc says: executes a SQL query and returns the number of rows affected, but affected_Rows calls sqlite3_total_changes() which only handles modifications (i.e. only INSERT, UPDATE or DELETE) and not the result of the query. A simple SELECT * table will return 0 even if the table is full to bursting. 2) Implementation has an impact on performance (see below).

Execute_Update is not implemented correctly for SQLite: since, after calling Execute_Query, we call Affected_Rows, we end up with the same problem.

7.2.2 Suggested corrections

Correcting these inconsistencies is not difficult, but the result will range from transparent to very slow (depending on the type of query and/or the volume of data returned, since the entire recordset have to be iterated).

Number_Of_Rows: As the name don't suggests, returns the number of rows in the recordset, not in the table. Can be implemented with the above reservations by iterating through the entire recordset.

Affected_Rows: you'll need to remember the last query performed by Execute_Update (in the connection record, next to UTF8_String). If this query contains INSERT, UPDATE or DELETE, we return the result unchanged; if it contains a SELECT *| Column_Name, we transform it into a SELECT COUNT(*|Column_Name). If the query contains anything else, we execute a standard Query with the above reservations, since we have to iterate through the entire recordset.

7.2.3 Suggested implementations

Two possibilities:
- Either I implement it at Gnoga level, but Pascal (Gnoga's maintainer) has to agree. It's an imperfect hack, but it'll be better than this unusable existing one.
- Or I can implement this in v22 and ban the use of Execute_Query, Execute_Update, Number_Of_Rows in the docs in favor of functions with identical names but prefixed v22.Sql, which is pretty dirty and wouldn't solve the problem of these rogue functions in Gnoga.Server.Database.

7.3 Add-ons

7.3.1 Mycelite

https://github.com/mycelial/mycelite

7.3.2 Compile time options

https://www.sqlite.org/compile.html

7.4 Concurrent access

7.4.1 Activating WAL mode

```
PRAGMA journal_mode=WAL;
sqlite3 github.db 'PRAGMA journal_mode=WAL;'
```

7.4.2 Managing busy error

- Setting delay (internal control)

  <<<TODO>>>

  https://www.sqlite.org/pragma.html#pragma_busy_timeout

- Call back function in Ada (external control)

  <<<TODO>>>

  https://www.sqlite.org/c3ref/busy_handler.html

7.4.3 Experimental

https://stackoverflow.com/questions/4060772/sqlite-concurrent-access

7.4.4 Production

SQLite can handle concurrent access in WAL (Write Ahead Log) mode. In this case all writes are appended to a WAL temporary fille which is periodically merged with the original database.

When SQLite is searching for something it would first check this temporary file and if nothing is found proceed with the main database file.

As a result, readers don't compete with writers and performance is much better compared to the traditional SQLite Rollback journaled mode.

https://www.sqlite.org/walformat.html
https://fly.io/blog/sqlite-internals-wal (How SQLite Scales Read Concurrency)

7.5 FAQ

7.5.1 Release history

https://www.sqlite.org/draft/changes.html

7.5.2 SQL virtual machine

https://fly.io/blog/sqlite-virtual-machine

7.5.3 Opening a new database connection

https://www.sqlite.org/c3ref/open.html
https://www.sqlite.org/uri.html
https://stackoverflow.com/questions/56416437/confusion-about-uri-path-to-configure-sqlite-database

7.5.4 UPSERT how to

- First pass (record does not exists)

  Update on a non existent record
  UPDATE System SET Value='0.0' WHERE Parameter='Shema_Version';
  Success : 0 record(s) affected

  Create ok because Changes() = 0 (no change from the last operation)
  INSERT INTO System (Parameter, Value) SELECT 'Shema_Version', '0.0' WHERE (Select Changes() = 0);
  Success : 1 record(s) affected

- Second pass (record already exists)

  Update an existent record successfull
  UPDATE System SET Value='0.1' WHERE Parameter='Shema_Version';
  Success : 1 record(s) affected

  No create since Changes() = 1 (last operation has change something)
  INSERT INTO System (Parameter, Value) SELECT 'Shema_Version', '0.1' WHERE (Select Changes() = 0);
  Success : 0 record(s) affected

7.6 Replication

7.6.1 Distributed SQLite

https://fly.io/docs/litefs
https://fly.io/docs/litefs/how-it-works
https://fly.io/blog/litefs-cloud
https://fly.io/blog/introducing-litefs
https://news.ycombinator.com/item?id=36602970

- Installing LiteFS without Docker

  https://github.com/superfly
  https://github.com/superfly/litefs/releases
  https://fly.io/docs/litefs/getting-started-fly

7.6.2 Cr-SQLite

https://github.com/vlcn-io/cr-sqlite

7.6.3 LiteStream

https://litestream.io
https://github.com/benbjohnson/litestream

https://fly.io/blog/all-in-on-sqlite-litestream
https://litestream.io/guides/systemd

7.7 Utilities

7.7.1 Sqlite-utils

https://sqlite-utils.datasette.io/en/stable/cli.html

7.7.2 SQLCrush

https://github.com/coffeeandscripts/sqlcrush

7.7.3 Visidata

https://www.visidata.org

```
user@system: sudo apt install visidata
```

7.8 Why SQLite ?

https://tailscale.com/blog/database-for-2022
https://simonwillison.net/2021/Jul/28/baked-data

8 Encryption

8.1 Key expansion

A SHA256 could be used as key expansion routine [from 32 to 256 bits]

https://ritul-patidar.medium.com/key-expansion-function-and-key-schedule-of-des-data-encryption-standard-algorithm-1bfc7476157

https://www.researchgate.net/figure/Structure-of-RC4-key-scheduling-process_fig8_267858656

8.2 RC4

geta : RC4 minimum key legnth

https://en.wikipedia.org/wiki/RC4
https://en.wikipedia.org/wiki/Stream_cipher
https://www.researchgate.net/publication/337743183_Modernized_RC4_encryption_algorithm/link/5de7eb71a6fdcc28370658dd/download
https://iopscience.iop.org/article/10.1088/1757-899X/420/1/012131
https://iopscience.iop.org/article/10.1088/1757-899X/680/1/012025

https://en.wikipedia.org/wiki/Variably_Modified_Permutation_Composition
https://microchipdeveloper.com/harmony:middleware-crypto

http://www.winpicprog.co.uk/pic_tutorial.htm

https://github.com/cforler/Ada-Crypto-Library

8.3 SHA-256

https://emn178.github.io/online-tools/sha256.html (interactive tool)

https://en.wikipedia.org/wiki/SHA-2
https://rosettacode.org/wiki/SHA-256

https://github.com/oilulio/Microcontroller-hashes/blob/master/sha256.c
https://www.mdpi.com/2071-1050/13/8/4324

9 TCP/IP

https://stackoverflow.com/questions/58361758/tcp-ip-using-ada-sockets-how-to-correctly-finish-a-packet

https://comp.lang.ada.narkive.com/V7LjubmR/tcp-ip-sockets-with-gnat-sockets

10 Unicode

10.1 Introduction

https://mcilloni.ovh/2023/07/23/unicode-is-hard

10.2 Glossary

Paragraphs are from theses Wikipedia links:
- https://en.wikipedia.org/wiki/Plane_(Unicode)
- https://en.wikipedia.org/wiki/Byte_order_mark
- https://en.wikipedia.org/wiki/Unicode
- https://en.wikipedia.org/wiki/UTF-8
- https://en.wikipedia.org/wiki/UTF-16

10.2.1 BMP

In the Unicode standard, a plane is a continuous group of 65,536 (216) code points. There are 17 planes, identified by the numbers 0 to 16, which corresponds with the possible values 00–1016 of the first two positions in six position hexadecimal format (U+hhhhhh).

Plane 0 is the Basic Multilingual Plane (BMP), which contains most commonly used characters.

10.2.2 BOM

BOM stands for Byte Order Mark. The BOM character is, simply, the Unicode code-point `U+FEFF ZERO WIDTH NO-BREAK SPACE`, encoded in the current encoding.

The encoded representation of the BOM depends of the representation. A text beginning with the sequence EF BB BF suggests an UTF-8 encoding.

UTF-8	EF BB BF
UTF-16 (BE)	FE FF
UTF-16 (LE)	FF FE

A BOM can signal several things to a program reading the text:

- The byte order, or endianness, of the text stream in the cases of 16-bit and 32-bit encodings;

- The fact that the text stream's encoding is Unicode, to a high level of confidence;

- Which Unicode character encoding is used.

BOM use is optional. Its presence interferes with the use of UTF-8 by software that does not expect non-ASCII bytes at the start of a file but that could otherwise handle the text stream.

According to the Unicode standard, the BOM for UTF-8 files is not recommended.

10.2.3 Unicode

Unicode, formally The Unicode Standard is an information technology standard for the consistent encoding, representation, and handling of text expressed in most of the world's writing systems.

Unicode can be stored using several different encodings, which translate the character codes into sequences of bytes. The Unicode Standard defines three encodings but several others exist, mostly variable-length encodings. The most common encodings are the ASCII-compatible UTF-8 and the ASCII-incompatible UTF-16, itself compatible with the obsolete UCS-2.

10.2.4 UTF-8

UTF-8 is a variable-length character encoding standard used for electronic communication. Defined by the Unicode Standard, the name is derived from Unicode (or Universal Coded Character Set) Transformation Format – 8-bit.

10.2.5 UTF-16

UTF-16 is a variable-length character encoding capable of encoding all 1,112,064 valid code points of Unicode (in fact this number of code points is dictated by the design of UTF-16). The encoding is variable-length, as code points are encoded with one or two 16-bit code units. UTF-16 arose from an earlier obsolete fixed-width 16-bit encoding, now known as UCS-2 (for 2-byte Universal Character Set), once it became clear that more than 216 (65,536) code points were needed.

UTF-16 is used by systems such as the Microsoft Windows API, the Java programming language and JavaScript/ECMAScript. It is used by SMS (the SMS standard specifies UCS-2, but almost all users actually implement UTF-16 so that emojis work).

UTF-16 is the only web-encoding that is incompatible with ASCII and never gained popularity on the web. The Web Hypertext Application Technology Working Group (WHATWG) considers UTF-8 "the mandatory encoding for all [text]" and that for security reasons browser applications should not use UTF-16.

UTF-16 BE stands for Big Endian and UTF-16 LE for Little Endian.

10.3 UXString

Paragraphs below are from the UXString author:
- https://github.com/Blady-Com/UXStrings

- compl.land.ada [ANN] Release of UXStrings 0.5.0 – 230505 at 05:23

10.3.1 Introduction

v22 adopts UXString v3 implementation using Unbounded_Wide_Wide_Strings for internal representation. With this latter implementation, the characters are stored in Unicode form and the management of dynamic size uses the standard Wide_Wide_Unbounded strings library.

Performance with Gnoga is better. UXStrings2 already brought better performance in the case of strings only made up of ASCII characters (improvement by a factor 2 to 3 compared to UXStrings1). With UXStrings3 performance in the latter case is still improved (factor 6 to 7 compared to UXStrings1) moreover in the case of strings accentuated in French and strings containing emojis the process times are also improved (factor 7 to 8 by compared to UXStrings1 or even more in the case of emojis). For all cases, the global memory occupation of the Gnoga application is generally similar.

10.3.2 Motivation

My first motivation was to avoid the user of the Ada language from having to make a choice in the representation of character strings. With the current Ada 2012 standard, the choice must be made according to the nature of the characters handled (Character, Wide_Character or Wide_Wide_Character) and the adaptation of the string size according to the operations carried out. Moreover, depending on the libraries used, making a single choice is generally not possible, which leads to continuous conversions.

Ada GUI library Gnoga internal character strings implementation is based on both Ada types String and Unbounded_String. The native Ada String encoding is Latin-1 whereas transactions with the Javascript part are in UTF-8 encoding.

Some drawbacks come up, for instance, with internationalization of programs (see Localize Gnoga demo):

- Several conversions between String and Unbounded_String objects it isn't usable out of Latin-1 character set, characters out of Latin-1 set are blanked;
- Continuous conversions between Latin-1 and UTF-8, each sent and received transaction between Ada and Javascript parts.

Two ways of possible improvement for native Ada String: dynamic length handling and Unicode support.

10.3.3 Definitions

```
Supported encoding schemes

type Encoding_Scheme is (ASCII_7, Latin_1, UTF_8, UTF_16BE, UTF_16LE);

Supported UTF-16 encoding schemes

subtype UTF_16_Encoding_Scheme is Encoding_Scheme range UTF_16BE .. UTF_16LE;

Characters in ISO/IEC 646

subtype ASCII_Character is Ada.Characters.Handling.ISO_646;
subtype ASCII_Character_Array is String with
```

```
      Dynamic_Predicate => (for all Item of ASCII_Character_Array => Item in ASCII_Character);

   Characters in ISO/IEC 8859-1

   subtype Latin_1_Character is Character;
   subtype Latin_1_Character_Array is String;

   Characters in Unicode Basic Multilingual Plane

   subtype BMP_Character is Wide_Character;
   subtype BMP_Character_Array is Wide_String;

   Characters in Unicode planes

   subtype Unicode_Character is Wide_Wide_Character;
   subtype Unicode_Character_Array is Wide_Wide_String;

   Array of 8 bits values representing UTF encodings (UTF-8, UTF-16BE, or UTF-16LE)

   subtype UTF_8_Character_Array is Ada.Strings.UTF_Encoding.UTF_String;
   subtype UTF_16_Character_Array is Ada.Strings.UTF_Encoding.UTF_String;

   Container type of Unicode characters with dynamic size usually named string

   type UXString is tagged private with
      Constant_Indexing => Element,
      Iterable => (First => First, Next => Next, Has_Element => Has_Element, Element => Element),
      String_Literal => From_Unicode;
```

10.3.4 Workarounds

First possibility is using UTF-8 as internal implementation in Unbounded_String objects. The simplest way but Gnoga uses many times character indexes to parse Javascript messages that is not easy to achieved with UTF-8 which may have several lengths to represent one character. String parsing will be time consuming. Some combinations may lead to incorrect UTF-8 representation.

Second possibility is to use Unbounded_Wide_String or Unbounded_Wide_Wide_String. Using Unbounded_Wide_String is quite being in the middle of the river might as well use Unbounded_Wide_Wide_String. In this latter case the memory penalty is heavy for only few accentuated character occurrences. So back to Unbounded_Wide_String but you'll miss the so essential emojis ;-)

Third possibility is to make no choice between Latin-1, Wide and Wide_Wide characters. The object shall adapt its inner implementation to the actual content. For instance with English language the most often use case will be Latin-1 inner implementation, for French language the most often will be Latin-1 with some exceptions with BMP (Unicode Basic Multilingual Plane) implementation such as in "cœur", for Greek language the most often will be BMP implementation. The programmer won't make any representation choice when for example receiving UTF-8 messages:

```
   S2 : UXString;
   ...
   S2 := "Received: " & From_UTF_8 (Message);
```

Automatically S2 will adapt its inner representation to the received characters.
UXStrings packages

Package named UXStrings (Unicode Extended Strings) and its Text_IO child package are proposed to bring String enhancements using some Ada 2022 features.

The first part of UXStrings package contains renaming statements of current Ada types. Ada current String type is structurally an array of Latin-1 characters thus is renamed as Latin_1_Character_Array. And so on.

The second part defines the USXString type as a tagged private type which has got aspects such as Constant_Indexing, Variable_Indexing, Iterable and String_Literal, so we can write:

```
   S1, S2, S3 : UXString;
   C          : Character;
   WC         : Wide_Character;
   WWC        : Wide_Wide_Character;
   ...
   S1 := "était blah blah";
   C   := S1 (3);
   WC  := S1 (2);
   WWC := S1 (1);
   S1 (3) := WWC;
   S1 (2) := WC;
   S1 (1) := C;
   S3  := "une soirée passée à étudier les mathématiques ℕℂ𝕂...";
   for I in S3 loop
      C   := S3 (I);
      WC  := S3 (I);
      WWC := S3 (I);
       Put_Line (Character'pos (C)'img & Wide_Character'pos (WC)'img & Wide_Wide_Char-
acter'pos (WWC)'img);
   end loop;
```

The third part defines conversion functions between UXString and various encoding such as Latin-1, BMP (USC-2), Unicode (USC-4), UTF-8 or UTF-16, so we can write:

```
   S1  := From_Latin_1 ("blah blah");
   S2  := From_BMP ("une soirée passée à étudier la physique ω=Δθ/Δt...");
   S3  := From_Unicode ("une soirée passée à étudier les mathématiques ℕℂ𝕂...");
   Send (To_UTF_8 (S1) & To_UTF_8 (S3));
```

The fourth part defines various API coming from Unbounded_String such as Append, "&", Slice, "=", Index and so on.

Note: Iterable is a GNAT specific aspect.

10.3.5 Some thoughts

I've preferred that the API of legacy Ada "string" types to be closed to those of Ada library so that the adaptation would be easy. Note that I've renamed them as character arrays rather than strings in order to accentuate the semantic difference.

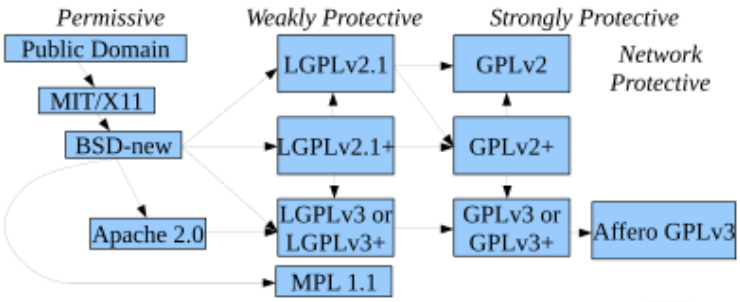
Apart from complex implementations, if you want to access a specific position you have to parse from the beginning of the UTF-8 data as UXStrings1 does. UXStrings2 always computes if the resulting data are all ASCII, so the access is then direct. UXStrings3 is internally like an Unicode array, so the access is direct.

Appendices

1 Copyrights & credits

1.1 Library Licence

v22 is copyright Sowebio under GPL v3 license.



1.1.1 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 [EUPL] 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 v22, a KISS library for Ada command line programs. Copyright ©2004, 2005, 2020, 2021, 2022, 2023 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 v22 Packages copyrights, credits and licences

Andreas Almroth: https://web.archive.org/web/20070403105909/http://www.almroth.com/adacurl [Curl binding, LGPL v2]
David Botton: https://www.linkedin.com/in/david-botton-3741b210 [Gnoga, LGPL v3]
Dmitry A. Kazakov: http://www.dmitry-kazakov.de/ada/components.htm [Simple Components, LGPLv2]
Jeffrey R. Carter: https://github.com/jrcarter/PragmARC [PragmAda Reusable Components, 3 Clause BSD]
Michael Rohan: https://sourceforge.net/projects/zanyblue [Internationalization, 3 Clause BSD]
Pascal Pignard: https://github.com/Blady-Com [UXStrings, CECILL v2.1]

2 To-do list

2.1 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/458719343aabd01cf-b17a3a4f7296797

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-pro-gram-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

2.2 Doc

2.2.1 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

5.1.1 GNAT CE 2019 - Exception with Delete_Tree dealing with broken symbolic links

Ada.Directories.Del_Tree crashes if a broken symbolic link in a directory of the tree to be deleted exists: raised ADA.IO_EXCEPTIONS.USE_ERROR: directory tree rooted at "/home/sr/opt/gnat-2019/lib/xmlada/xmlada_input.relocatable" could not be deleted

- Demo

```
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

Around More_Entries > Fetch_Next_Entry > readdir_gnat > Match

Here a broken symbolic link libxmlada_input_sources.so which is declared “non exis-
tent” by File_Exists_Attr [C_Full_Name'Address,Attr'Access]; en 776 which is
__gnat_file_exists_attr in 1668 of adaint.c with a reference to a structure in adain-
t.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;
-----------------------------

Who calls  *__gnat_stat_to_attr* and test a file descriptor at -1, [broken symbolic
link I guess]. Then __gnat_stat returns 2 to __gnat_stat_to_att with a test at line
1124 of adaint.c

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

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

<shadok> So if you can't find the file, there's no error. </shadok>
```

- Conclusion

  The rest becomes clear... The broken symbolic link libxmlada_input_sources.so is declared to not exist, the routine exits of the current directory [which it is believed to be empty, to delete it, and then crashes when it tries to erase this directory, which is empty but isn't...

- Solving

  We could re-code this recursive function in a simpler way. However, the best thing to do would be to correct the anomaly which is probably in _gnat_stat, so that this function returns the correct value and doesn't confuse 'doesn't exist' [the file to which the symbolic link points] with 'doesn't exist' [the symbolic file].

Ada, « it's stronger than you ».
Tribute to Daniel Feneuille, a legendary french Ada teacher [and much more][6]

The link below is kept here for future use...

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

[6] http://d.feneuille.free.fr