Report concerning:

**Program Initation**

By

**L. Dries**

dd. **16-08-2019**

Version: 1.00 Original version 16-08-2019 L. Dries

# Index

[1 Index 3](#_Toc16440967)

[2 Introduction 4](#_Toc16440968)

[3 General 5](#_Toc16440969)

[4 File structure 6](#_Toc16440970)

[5 Use of the package 7](#_Toc16440971)

[6 Listings 9](#_Toc16440972)

[6.1 Specification 9](#_Toc16440973)

[6.2 Body 12](#_Toc16440974)

# Introduction

In this report a description is give of the ADA package “init”.

In programs there is sometimes a need for initialization of parameters within the program. In many cases this values must be kept the same after restart of that program, but not everybody wants the same initialization values, for instance if the program can run present its interaction in different languages it is understandable that one user will have English conversation and another will have German conversation.

In early versions of Microsoft Windows (3.0 and 3.1) the operating system gave a possibility to preserve these parameters in a file coupled to the program with the extension .INI .

We would like the same facility but with a further extension that some sets of parameters could be used in more programs. The package must also be usable on different operating systems and computer systems while the contents of the file must be transportable between all of those systems

The package “ init” must offer such a facility.

# General

The package Init create a system that is able to create a number of files that contain data usable over various instances of the program. The program will when started read a number of such files and if they do not exist on the defined position it will create them using standard initiation values. Such as in the example of the previous chapter an initiation language “English”. Internally in the program there will a data base where al initializations are kept. If an initialization is changed the file will be altered. The data base will be updated regularly

# File structure

The file ia a complete ASCII file in which has the following structure.

|  |  |  |  |
| --- | --- | --- | --- |
| **Line** | **Contents** | **Example** |  |
| 1 | “INI” | “INI” | Identification |
| 2 | Version nr | 1 | Version of file |
| C1 | “CAT” | “CAT” | Start of a category |
| C2 | Category name | Languages | Name of a category |
| E1 | Element name | “Lan” | Name of en element |
| E2 | Element value | “GB-USA” | Value of the element |
| last | “EOF” | “EOF” | End of file |

Every category has the lines C1 and C2 and at least one set of E1 and E2 lines. A category can have more Elements.

The file can be extended at any moment and if the program does not need an element in some configuration of the program that is no problem

# Use of the package

The package has the following:

function use\_file(Name : in string) return integer;

The filename is a string without the file-extension “.ini” and can be in two formats

1. Only a name. The file is then positioned in the same directory as the executable file of the program. In general these files are used for that program only
2. A name that consists of the complete path. These files are in general used if they are needed in several programs.

The functions returns an integer consisting of a number identifying that file during the rest of the program.

procedure Set\_Value( ID : in integer; Cat\_Name : in string;

Val\_Name : in string; Val : in string);

procedure Set\_Value( ID : in integer; Cat\_Name : in string;

Val\_Name : in string; Val : in integer);

procedure Set\_Value( ID : in integer; Cat\_Name : in string;

Val\_Name : in string; Val : in long\_integer);

procedure Set\_Value( ID : in integer; Cat\_Name : in string;

Val\_Name : in string; Val : in float);

procedure Set\_Value( ID : in integer; Cat\_Name : in string;

Val\_Name : in string; Val : in boolean);

If you only have to set a value in the initiation file.

ID is then the number returned by use\_file.

Cat\_Name is the name of the category.

Val\_Name is the name of the value.

Val is the value you want to put in

An example could be Cat\_Name := “Dimension”

Val\_Name := “Length”

Val := 10

With a second one Cat\_Name := “Dimension”

Val\_Name := “Width”

Val := 5

function Get\_Value( ID : in integer; Cat\_Name : in string;

Val\_Name : in string; Default\_Value : in string) return string;

function Get\_Value( ID : in integer; Cat\_Name : in string;

Val\_Name : in string; Default\_Value : in integer) return integer;

function Get\_Value( ID : in integer; Cat\_Name : in string;

Val\_Name : in string; Default\_Value : in long\_integer)

return long\_integer;

function Get\_Value( ID : in integer; Cat\_Name : in string;

Val\_Name : in string; Default\_Value : in float) return float;

function Get\_Value( ID : in integer; Cat\_Name : in string;

Val\_Name : in string; Default\_Value : in boolean) return boolean;

If you are asking a value from the database you use one of these functions

The meaning of the identifiers is the same as in the Set\_Value functions and then added

Default\_Value that is the default value to be taken when the combination Cat\_Name and Val\_Name do not exist in the database

procedure Close\_ini;

This function must be executed just before the your program ends to save the ini-file. Be careful that if your program ends by an error it is you that must decide if the ini-file must also be saved

# Listings

## Specification

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

-- Program\_Init (V1.00) A package to create INI files --

-- --

-- Copyright (C) 15-4-2010 L. Dries --

-- --

-- This library is free software; you can redistribute it and/or --

-- modify it under the terms of the GNU General Public --

-- License as published by the Free Software Foundation; either --

-- version 2 of the License, or (at your option) any later version. --

-- --

-- This library is distributed in the hope that it will be useful, --

-- but WITHOUT ANY WARRANTY; without even the implied warranty of --

-- MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU --

-- General Public License for more details. --

-- --

-- You should have received a copy of the GNU General Public --

-- License along with this library; if not, write to the --

-- Free Software Foundation, Inc., 59 Temple Place - Suite 330, --

-- Boston, MA 02111-1307, USA. --

-- --

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

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

-- This Library creates the possibility to use initiation files for --

-- ADA programs. Multiple files are possible. The purpose of INI --

-- files is to save values used in a program or a number of programs --

-- to use the next time when one of these programs are used again. --

-- As an example: A program needs a file with general information. --

-- When is not practical to put the location in the program it can be--

-- pratctical if the first time the location is asked and next times --

-- the program just knows where to look. -–

-- --

-- Two alternate locations for the INI files can be created: --

-- 1: The home directory of the program (std) --

-- 2: A location in another directory --

-- The first location is in general for the program itself, the --

-- can be used for INI files that can be used by various programs, --

-- for instance the location of a file containing conversion factors --

-- for dimensions say from km to miles etc. --

-- --

-- V 1.00 L. Dries, Rotterdam, The Netherlands 15-05-2009 --

-- Original --

-- V 1.01 L. Dries, Rotterdam, The Netherlands 29-07-2016 --

-- Booleans Added --

-- V 1.02 L. Dries, Rotterdam, The Netherlands 07-05-2018 --

-- Long Integers added --

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

with Ada.Strings.Unbounded; use Ada.Strings.Unbounded;

-- with Unchecked\_Deallocation;

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

-- --

-- The GNAT.OS\_Lib package is used to make the package as operating --

-- system independent as possible --

-- --

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

with GNAT.OS\_Lib;

package Program\_Init is

type Waarde is limited private;

type Waarde\_Pointer is access Waarde;

type Categorie is limited private;

type Categorie\_Pointer is access Categorie;

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

-- --

-- The Record File\_ID consists of the name of the file (without the --

-- extension, an File ID number, a boolean std for the location of the --

-- directory and the start of a list of categories. The record is --

-- organized as a list --

-- --

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

type File\_ID;

type File\_Pointer is access File\_ID;

type File\_ID is

record

ID : integer := -1;

name : Unbounded\_string;

cat : Categorie\_Pointer := null;

next : File\_Pointer := null;

end record;

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

-- --

-- To create an operating system independent filename the directory --

-- separator comes from then GNAT.OS\_Lib package --

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

DS : character := GNAT.OS\_Lib.Directory\_Separator;

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

-- The function which defines the name and the type of the INI file --

-- used. --

-- Name The name of the file (without its .INI extension --

-- in the case standard is false The complete file name --

-- including the directories where the file is located must --

-- be given --

-- The return value is a file identifier later on used in the program to --

-- get or set variables --

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

function use\_file(Name : in string) return integer;

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

-- --

-- A procedure which sets for a given Category and Value name a value --

-- The type of values to be used are string, integer or float --

-- ID idenrifies the file --

-- Cat\_Name Name of the catagory --

-- Val\_Name Name of the value --

-- Val Value to be set --

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

procedure Set\_Value( ID : in integer; Cat\_Name : in string;

Val\_Name : in string; Val : in string);

procedure Set\_Value( ID : in integer; Cat\_Name : in string;

Val\_Name : in string; Val : in integer);

procedure Set\_Value( ID : in integer; Cat\_Name : in string;

Val\_Name : in string; Val : in long\_integer);

procedure Set\_Value( : in integer; Cat\_Name : in string;

Val\_Name : in string; Val : in float);

procedure Set\_Value( ID : in integer; Cat\_Name : in string;

Val\_Name : in string; Val : in boolean);

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

-- --

-- A which returns for a given Category and Value name a value --

-- The types of values to be used are string, integer or float --

-- If a the Category or the Value does not exists the function returns the --

-- default value and sets the default value --

-- ID idenrifies the file --

-- Cat\_Name Name of the catagory --

-- Val\_Name Name of the value --

-- Default\_Value Value to be set --

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

function Get\_Value( ID : in integer; Cat\_Name : in string;

Val\_Name : in string; Default\_Value : in string) return string;

function Get\_Value( ID : in integer; Cat\_Name : in string;

Val\_Name : in string; Default\_Value : in integer) return integer;

function Get\_Value( ID : in integer; Cat\_Name : in string;

Val\_Name : in string; Default\_Value : in long\_integer)

return long\_integer;

function Get\_Value( ID : in integer; Cat\_Name : in string;

Val\_Name : in string; Default\_Value : in float) return float;

function Get\_Value( ID : in integer; Cat\_Name : in string;

Val\_Name : in string; Default\_Value : in boolean) return boolean;

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

-- --

-- A procedure which creates a new INI file with the values inserted in --

-- it --

-- If a the Category of the Value do not exists the function returns the --

-- default value and sets the default value --

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

procedure Close\_ini;

private

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

-- --

-- The Record "Waarde" (Dutch word for Value) consits of the name --

-- of the value and the value itself. This value is saved in the --

-- format of an unbounded string. The record is organized as at --

-- list --

-- --

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

type Waarde is

record

Name : Unbounded\_string;

Waarde : Unbounded\_String;

next : Waarde\_Pointer;

end record;

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

-- --

-- The Record "Categorie" (Dutch word for Category) consists of the name --

-- of the category and the start of a list of Waarde records. The record --

-- is organized as a list --

-- --

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

type Categorie is

record

Name : Unbounded\_string;

waarde : Waarde\_Pointer := null;

next : Categorie\_Pointer := null;

end record;

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

-- --

-- Startpoint for the files list --

-- --

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

Start\_File : File\_Pointer := null;

engine : constant integer := 1;

end Program\_Init;

## Body

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

-- Program\_Init A package to create INI files --

-- --

-- Copyright (C) 2009 L. Dries --

-- --

-- This library is free software; you can redistribute it and/or --

-- modify it under the terms of the GNU General Public --

-- License as published by the Free Software Foundation; either --

-- version 3 of the License, or (at your option) any later version. --

-- --

-- This library is distributed in the hope that it will be useful, --

-- but WITHOUT ANY WARRANTY; without even the implied warranty of --

-- MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU --

-- General Public License for more details. --

-- --

-- You should have received a copy of the GNU General Public --

-- License along with this library; if not, write to the --

-- Free Software Foundation, Inc., 59 Temple Place - Suite 330, --

-- Boston, MA 02111-1307, USA. --

-- --

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

with Program\_Init; use Program\_Init;

with Ada.Strings.Unbounded; use Ada.Strings.Unbounded;

with Text\_IO; use Text\_IO;

with Unchecked\_Deallocation;

package body Program\_Init is

-- procedure Free\_File\_Pointer is new Unchecked\_Deallocation(File\_ID, File\_Pointer);

function use\_file(Name : in string) return integer is

tFile\_ID : File\_pointer;

Ctemp : Categorie\_Pointer;

Wtemp : Waarde\_Pointer;

Ini\_File : File\_Type;

n : integer;

nr : integer;

eof : boolean := false;

File\_Name : Unbounded\_string;

In\_String : Unbounded\_string;

begin

tFile\_ID := Start\_File;

n := 1;

if tFile\_ID = null then

tFile\_ID := new File\_ID;

else

while tFile\_ID.next /= null loop

tFile\_ID := tFile\_ID.next;

n := tFile\_ID.ID;

end loop;

n := n + 1;

tFile\_ID.next := new File\_ID;

tFile\_ID := tFile\_ID.next;

end if;

tFile\_ID.ID := n;

tFile\_ID.name := To\_Unbounded\_String(Name);

File\_Name := To\_Unbounded\_String(Name & ".ini");

Open ( Ini\_File, In\_File, To\_String(File\_Name));

In\_String := To\_Unbounded\_String(Get\_Line (Ini\_File));

if To\_String(In\_String) = "INI" then

nr := integer'value(Get\_Line(Ini\_File));

if nr <= 1 then

while not eof loop

In\_String := To\_Unbounded\_String(Get\_Line (Ini\_File));

eof := To\_String(In\_String) = "EOF";

if not eof then

if In\_String = "CAT" then

if tFile\_ID.cat = null then

tFile\_ID.cat := new Categorie;

Ctemp := tFile\_ID.cat;

else

Ctemp.next := new Categorie;

Ctemp := Ctemp.next;

end if;

In\_String := To\_Unbounded\_String(Get\_Line (Ini\_File));

Ctemp.Name := In\_String;

In\_String := To\_Unbounded\_String(Get\_Line (Ini\_File));

end if;

if In\_String = "VAL" then

if Ctemp.waarde = null then

Ctemp.waarde := new Waarde;

Wtemp := Ctemp.waarde;

else

Wtemp.next := new Waarde;

Wtemp := Wtemp.next;

end if;

In\_String := To\_Unbounded\_String(Get\_Line (Ini\_File));

Wtemp.Name := In\_String;

In\_String := To\_Unbounded\_String(Get\_Line (Ini\_File));

Wtemp.Waarde := In\_String;

end if;

end if;

end loop;

else

Close( Ini\_File );

raise CONSTRAINT\_ERROR;

end if;

else

Close ( Ini\_File );

raise NAME\_ERROR;

end if;

Close ( Ini\_File );

if n = 1 then

Start\_File := tFile\_ID;

end if;

return n;

exception

when NAME\_ERROR =>

if n = 1 then

Start\_File := tFile\_ID;

end if;

return n;

end use\_file;

function Get\_File( ID : in integer) return File\_Pointer is

tFile\_ID : File\_Pointer := Start\_File;

begin

if tFile\_ID /= null then

loop

exit when tFile\_ID.ID = ID;

tFile\_ID := tFile\_ID.next;

exit when tFile\_ID = null;

end loop;

end if;

return tFile\_ID;

end Get\_File;

function Get\_Cat ( tFile\_ID : in File\_Pointer; Cat\_Name : string) return Categorie\_Pointer is

tCat : Categorie\_Pointer := tFile\_ID.cat;

begin

if tCat /= null then

loop

exit when To\_String(tCat.Name) = Cat\_Name;

tCat := tCat.next;

exit when tCat = null;

end loop;

end if;

return tCat;

end Get\_Cat;

function Get\_Waarde ( tCat : in Categorie\_Pointer; Val\_Name : string) return Waarde\_Pointer is

tVal : Waarde\_Pointer := tCat.waarde;

begin

if tVal /= null then

loop

exit when To\_String(tVal.Name) = Val\_Name;

tVal := tVal.next;

exit when tVal = null;

end loop;

end if;

return tVal;

end Get\_Waarde;

procedure Set\_Value( ID : in integer; Cat\_Name : in string; Val\_Name : in string; Val : in string) is

tFile\_ID : File\_Pointer;

tCat : Categorie\_Pointer;

tVal : Waarde\_Pointer;

begin

tFile\_ID := Get\_File ( ID );

if tFile\_ID /= null then

tCat := Get\_Cat(tFile\_ID, Cat\_Name);

if tCat = null then

if tFile\_ID.cat = null then

tFile\_ID.cat := new Categorie;

tCat := tFile\_ID.cat;

else

tCat := tFile\_ID.cat;

while tCat.next /= null loop

tCat := tCat.next;

end loop;

tCat.next := new Categorie;

tCat := tCat.next;

end if;

tCat.Name := To\_Unbounded\_String(Cat\_Name);

end if;

tVal := Get\_Waarde ( tCat, Val\_Name);

if tVal = null then

if tCat.waarde = null then

tCat.waarde := new Waarde;

tVal := tCat.waarde;

else

tVal := tCat.waarde;

while tVal.next /= null loop

tVal := tVal.next;

end loop;

tVal.next := new Waarde;

tVal := tVal.next;

end if;

tVal.Name := To\_Unbounded\_String(Val\_Name);

end if;

tVal.Waarde := To\_Unbounded\_String(Val);

end if;

end Set\_Value;

procedure Set\_Value( ID : in integer; Cat\_Name : in string; Val\_Name : in string; Val : in integer) is

begin

Set\_Value(ID, Cat\_Name, Val\_Name, Integer'image(Val));

end Set\_Value;

procedure Set\_Value( ID : in integer; Cat\_Name : in string; Val\_Name : in string; Val : in long\_integer) is

begin

Set\_Value(ID, Cat\_Name, Val\_Name, long\_integer'image(Val));

end Set\_Value;

procedure Set\_Value( ID : in integer; Cat\_Name : in string; Val\_Name : in string; Val : in float) is

begin

Set\_Value(ID, Cat\_Name, Val\_Name, Float'image(Val));

end Set\_Value;

procedure Set\_Value( ID : in integer; Cat\_Name : in string; Val\_Name : in string; Val : in boolean) is

begin

Set\_Value(ID, Cat\_Name, Val\_Name, boolean'image(Val));

end Set\_Value;

function Get\_Value( ID : in integer; Cat\_Name : in string; Val\_Name : in string; Default\_Value : in string) return string is

Value : unbounded\_string;

tFile\_ID : File\_Pointer;

tCat : Categorie\_Pointer;

tVal : Waarde\_Pointer;

exist : boolean := false;

begin

Value := To\_Unbounded\_string(Default\_Value);

tFile\_ID := Get\_File ( ID );

if tFile\_ID /= null then

tCat := Get\_Cat(tFile\_ID, Cat\_Name);

if tCat /= null then

tVal := Get\_Waarde ( tCat, Val\_Name);

if tVal /= null then

Value := tVal.Waarde;

exist := true;

end if;

end if;

end if;

if not exist then

Set\_Value( ID, Cat\_Name, Val\_Name, Default\_Value );

end if;

return To\_String(Value);

end Get\_Value;

function Get\_Value( ID : in integer; Cat\_Name : in string; Val\_Name : in string; Default\_Value : in integer) return integer is

Value : integer;

begin

Value := integer'value(Get\_Value( ID, Cat\_Name, Val\_Name, Integer'image(Default\_Value)));

return Value;

end Get\_Value;

function Get\_Value( ID : in integer; Cat\_Name : in string; Val\_Name : in string; Default\_Value : in long\_integer) return long\_integer is

Value : long\_integer;

begin

Value := long\_integer'value(Get\_Value( ID, Cat\_Name, Val\_Name, long\_Integer'image(Default\_Value)));

return Value;

end Get\_Value;

function Get\_Value( ID : in integer; Cat\_Name : in string; Val\_Name : in string; Default\_Value : in float) return float is

Value : float;

begin

Value := float'value(Get\_Value( ID, Cat\_Name, Val\_Name, float'image(Default\_Value)));

return Value;

end Get\_Value;

function Get\_Value( ID : in integer; Cat\_Name : in string; Val\_Name : in string; Default\_Value : in boolean) return boolean is

Value : boolean;

begin

Value := boolean'value(Get\_Value( ID, Cat\_Name, Val\_Name, boolean'image(Default\_Value)));

return Value;

end Get\_Value;

procedure Close\_Ini is

Ini\_File : File\_Type;

tFile\_ID : File\_pointer;

temp : File\_Pointer := null;

Ctemp : Categorie\_Pointer;

Wtemp : Waarde\_Pointer;

File\_Name : Unbounded\_string;

Out\_String : Unbounded\_string;

begin

tFile\_ID := Start\_File;

while tFile\_ID /= null loop

temp := tFile\_ID;

tFile\_ID := tFile\_ID.next;

if temp /= null then

File\_Name := temp.name & ".ini";

Create ( Ini\_File, Out\_File, To\_String(File\_Name));

Put\_Line (Ini\_File, "INI");

Put\_Line (Ini\_File, Integer'image(engine));

Ctemp := temp.cat;

while Ctemp /= null loop

Put\_Line ( Ini\_File, "CAT" );

Put\_Line ( Ini\_File, To\_String(Ctemp.Name));

Wtemp := Ctemp.waarde;

while Wtemp /= null loop

Put\_Line ( Ini\_File, "VAL" );

Put\_Line ( Ini\_File, To\_String(Wtemp.Name));

Put\_Line ( Ini\_File, To\_String(Wtemp.Waarde));

Wtemp := wTemp.next;

end loop;

Ctemp := Ctemp.next;

end loop;

Put\_Line (Ini\_File, "EOF");

Close ( Ini\_File );

-- Free\_File\_Pointer(temp);

end if;

end loop;

end Close\_Ini;

end Program\_Init;