



Thistle Type A Interface for Persistent Variables
Version 1

CML00080-01

Code Magus Limited (England reg. no. 4024745)
Number 6, 69 Woodstock Road
Oxford, OX2 6EY, United Kingdom
www.codemagus.com
Copyright © 2014 by Code Magus Limited
All rights reserved



December 15, 2020

Contents

1	Thistle Interface	1
1.1	Introduction	1
1.2	Thistle Interface Definition for PSTORE	1
2	PSTORE Type A Interface reference	2
2.1	Interface Methods	2
2.1.1	error()	2
2.1.2	open()	2
2.1.3	close()	3
2.1.4	assist()	3
2.1.5	getvar()	3
2.1.6	setvar()	4
2.1.7	unsetvar()	5
2.1.8	list_children()	5
2.2	Example Thistle Script	5

1 Thistle Interface

1.1 Introduction

The Code Magus Limited *Persistent Store* allows an application to store and manipulate named variables in a manner that makes them persistent across different invocations of the application and even across different applications.

The Thistle Type A Interface `PSTORE` is the interface to the Code Magus *Persistent Store* Library and provides the means by which Thistle scripts (packages and usecases) can store and retrieve named variables.

This document shows how persistent variables can be set, retrieved, unset and listed by Thistle scripts.

1.2 Thistle Interface Definition for PSTORE

.

The Thistle Interface Definition for the *Persistent Store* is a normal Type A Interface to Thistle.

```
interface PSTORE;
{
  -- $Author: patrick $
  -- $Date: 2014/07/09 09:47:16 $
  -- $Id: PSTORE.tid,v 1.4 2014/07/09 09:47:16 patrick Exp $
  -- $Name: $
  -- $Revision: 1.4 $
}
```

```
-- $State: Exp $
--
-- $Log: PSTORE.tid,v $
-- Revision 1.4  2014/07/09 09:47:16  patrick
-- tid file packaged in bin
--
-- Revision 1.3  2012/07/20 12:28:10  justin
-- Fixed for windows.
--
-- Revision 1.2  2011/02/20 17:56:30  hayward
-- Changed for use on Linux.
--
-- Revision 1.1.1.1  2011/02/18 14:51:38  hayward
-- Import sources to CVS.
--
}
    type : typea;
    module : "${THISTLEPORTALBINPATH}pstoretai${THISTLEPORTALSUFDL}";
    init : pstoretai_init;
end.
```

2 PSTORE Type A Interface reference

2.1 Interface Methods

The following methods are exposed by the PSTORE Type A Interface.

2.1.1 error()

`error()` Function `error()` is the Type A Interface wrapper function for the *Persistent Store* library function `pstore_error()`. This function returns the last error string from the pstore library to the caller. The function also prints the error string to the Thistle log.

Parameters for Function `error()`

`pstore` The `pstore` parameter is the value returned by the `open()` function and the handle on which all *Persistent Store* operations are performed.

2.1.2 open()

`open()` Function `open()` is the Type A Interface wrapper function for the *Persistent Store* library function `pstore_open()`.

This function maps between the `Thistle` called function and the `pstore` library function. If the `pstore` library operation fails then this writes the error message to the log and reports the failure to the executing script.

If function `open()` successfully opens a *Persistent Store* instance, then the function returns a blob which is used as a handle for operations on the *Persistent Store*. This blob must be passed back to all other library function in order to perform the respective operation on the `pstore`.

Parameters for Function `open()`

- `host` `host` is a required parameter and is either an Internet protocol (IP) address in dotted decimal form (as in 127.0.0.1) or a domain name address (as in www.codemagus.com). This names the host machine on which the *Persistent Store* resides.
- `port` The port parameter is optional and specifies which IP port on the host to connect to. There may be more than one instance of the *Persistent Store* server running on a machine and supplying different name spaces of variables. If this option is not specified then the default of 60060 is used.
- `flags` The flags parameter is optional and supplies, in operator-string form (see `flagopts.pdf`), the values of the desired flags. The following flags are supported:

Flag Value	Flag String	Option
<code>PSTORE_OPT_VERBOSE</code>	<code>VERBOSE</code>	process with maximum message output

Examples

```
flags:="/VERBOSE"
flags:="/NONE"
```

2.1.3 `close()`

`close()` Function `close()` is the wrapper function for the `pstore` function `pstore_close()`.

This function maps between the `Thistle` called function and the *Persistent Store* library function. If the `pstore` library operation fails then this writes the error message to the log and reports the failure to the executing script.

Parameters for Function `close()`

- `pstore` The `pstore` parameter is the value returned by the `open()` function and the handle on which all *Persistent Store* operations are performed.

2.1.4 `assist()`

`assist()` Function `assist()` returns help text about the *Persistent Store* system.

Parameters for Function `assist()`

The `assist()` function has no parameters.

2.1.5 `getvar()`

`getvar()` Function `getvar()` wraps the `pstore_getvar()` function.

This function maps between the `Thistle` called function and the `pstore` library function. If the `pstore` library operation fails then this writes the error message to the log and reports the failure to the executing script.

Parameters for Function `getvar()`

`pstore` The `pstore` parameter is the value returned by the `open()` function and the handle on which all *Persistent Store* operations are performed.

`name` The `name` parameter is the name of the variable for which the value must be retrieved.

The return value from `getvar()` is a `Thistle` variable with the type set to the type the variable was originally set with.

2.1.6 `setvar()`

`setvar()` Function `setvar()` wraps the `pstore_setvar()` function.

This function maps between the `Thistle` called function and the `pstore` library function. If the `pstore` library operation fails then this writes the error message to the log and reports the failure to the executing script.

Parameters for Function `setvar()`

`pstore` The `pstore` parameter is the value returned by the `open()` function and the handle on which all *Persistent Store* operations are performed.

`name` The `name` parameter is the name of the variable for which the value must be retrieved.

`buffer` The `buffer` parameter is where the value of the named parameter should be set before calling this function. The buffer must be a BLOB type.

`type` The `type` parameter is optional and specifies what type of the data in the buffer and to which of variable must be set to by the *Persistent Store* system. Although the buffer must be a BLOB the actual contents can be either a string or a BLOB. If this parameter is not specified then the default of string is used.

Examples

```
type := "/STRING"
type := "/BLOB"
```

The return value from `setvar()` is a zero if the call was successful.

2.1.7 unsetvar()

`unsetvar()` Function `unsetvar()` wraps the `pstore_unsetvar()` function.

This function maps between the `Thistle` called function and the `pstore` library function. If the `pstore` library operation fails then this writes the error message to the log and reports the failure to the executing script.

Parameters for Function `unsetvar()`

`pstore` The `pstore` parameter is the value returned by the `open()` function and the handle on which all *Persistent Store* operations are performed.

`name` The `name` parameter is the name of the variable that is to be unset.

The return value from `setvar()` is a zero if the call was successful.

2.1.8 list_children()

`list_children()` Function `list_children()` wraps the `pstore_list_children()` function.

This function maps between the `Thistle` called function and the `pstore` library function. If the `pstore` library operation fails then this writes the error message to the log and reports the failure to the executing script.

Parameters for Function `list_children()`

`pstore` The `pstore` parameter is the value returned by the `open()` function and the handle on which all *Persistent Store* operations are performed.

`name` The name parameter is the name of the variable node for which the children nodes or variable names should be returned.

The return value from `list_children()` is a Thistle variable with type string and the value is a comma separated list of names.

2.2 Example Thistle Script

The module `pstoretai` wraps the `pstoretai` library interface and exposes it to the Thistle environment as a Type A Interface. The following Thistle Interface Definition defines this interface to the Thistle environment:

```
interface PSTORE;
{
  -- $Author: patrick $
  -- $Date: 2014/07/09 09:47:16 $
  -- $Id: PSTORE.tid,v 1.4 2014/07/09 09:47:16 patrick Exp $
  -- $Name: $
  -- $Revision: 1.4 $
  -- $State: Exp $
  --
  -- $Log: PSTORE.tid,v $
  -- Revision 1.4 2014/07/09 09:47:16 patrick
  -- tid file packaged in bin
  --
  -- Revision 1.3 2012/07/20 12:28:10 justin
  -- Fixed for windows.
  --
  -- Revision 1.2 2011/02/20 17:56:30 hayward
  -- Changed for use on Linux.
  --
  -- Revision 1.1.1.1 2011/02/18 14:51:38 hayward
  -- Import sources to CVS.
  --
}
  type : typea;
  module : "${THISTLEPORTALBINPATH}pstoretai${THISTLEPORTALSUFDL}";
  init : pstoretai_init;
end.
```

The Thistle run-time locates the named interface definition using the standard Thistle external pathing convention. For example, the following in the preamble of a script

```
interface Pstore : CodeMagusExtras.PSTORE;
```

introduces `pstore` as an internal local name of the interface. This expects that the variable `CodeMagusExtras` is defined suitably in a Thistle configuration file. For example in the directory containing a script with the above interface definition in its preamble, a configuration file might include:

2.2 Example Thistle Script 2 PSTORE TYPE A INTERFACE REFERENCE

```
DefaultRootDirectory=C:\
CodeMagusExtras=Codemagus\CodeMagus\bin\
```

and in which case, the Thistle interface definition file shown above would be expected to have the fully qualified name of

```
C:\CodeMagus\CodeMagus\bin\PSTORE.tid
```

The following script shows how the PSTORE interface could be used for processing persistent variables:

```
package PSTORESamplePackage;

    created by 'Code Magus Limited';
    description 'Test PSTORE TypeA Interface';
    date          2011-02-18T14:00:00;
    target        'Persistent Sore';

    interface PStore          : CodeMagus.PSTORE;
begin
    cmstore := PStore.Connect();

    cmstore_host := "195.137.70.68"; {Patrick H.}
    cmstore_host := "localhost"; {Patrick H.}
    cmstore_port := "60060";

    cmstore_h := cmstore.open(cmstore_host, cmstore_port);

    testvalue := System.TranslateFromStringToASCII("hello");
    testvalue2 := System.TranslateFromStringToASCII("hello2");

    cmstore.setvar(cmstore_h, "test.nameblob", testvalue, "+BLOB");
    cmstore.setvar(cmstore_h, "test.namestring", testvalue2, "+STRING");

    tmp1 := cmstore.getvar(cmstore_h, "test.nameblob");
    System.DumpScope(tmp1);

    tmp2 := cmstore.getvar(cmstore_h, "test.namestring");
    System.DumpScope(tmp2);

    tmp3 := cmstore.list_children(cmstore_h, "test");
    System.WriteLine("List value is : " # tmp3);

    cmstore.close(cmstore_h);

end.
```

Running this script produces the expected output:

```
._output.txt
```