CODE MAGUS

RECIO: Thistle Type A Interface to the Code Magus Record Stream I/O Library

CML00021-01

Code Magus Limited (England reg. no. 4024745)
Number 6, 69 Woodstock Road
Oxford, OX2 6EY, United Kingdom
www.codemagus.com

Business Partner IBM

August 16, 2016

Contents

1 Thistle Interface

1.1 Introduction

The Code Magus Limited Record Stream I/O Library provides a generic interface that has the ability to specialised separately to satisfy various requirements and environments. The library provides a separate interface for the implementation of these specialisations whose details are supplied by a configuration file in each case. See the document `recio.pdf` for details of the Record Stream I/O Library.

The *Thistle* Type A Interface `RECIO` is the interface to the Code Magus Record Stream I/O Library and provides the means by which *Thistle* scripts (packages and usecases) can process record files using the access methods of the Record Stream I/O Library. There should not be a restriction in the access methods available to *Thistle* scripts as opposed to those access methods available other tools that use the `RECIO` library for processing records.

This document shows how the access `RECIO` access methods can be used by the *Thistle* scripts. For the available access methods and for the options and modes supported by the individual access methods, the corresponding access method documentation should be used. This document uses the `TEXT` access method to illustrate how calls are made in a *Thistle* script as opposed another high-level language written tool that would use the access method.

1.2 *Thistle* Interface Definition for **RECIO**

The *Thistle* Interface Definition for RECIO is a normal Type A Interface to *Thistle*.

```
interface RECIO;

{
  -- File: RECIO.tid
  --
  -- This is the Thistle Interface Definition for the Type A Interface to the
  -- Code Magus Limited Record Stream I/O Library. This interface makes the
  -- Record I/O Access Methods available to Thistle Scripts.
  --
  -- Copyright (c) 2008 by Code Magus Limited. All rights reserved.
  --
  -- Author: Stephen R. Donaldson [stephen@codemagus.com].
  --

  -- $Author: patrick $
  -- $Date: 2014/07/08 12:10:15 $
  -- $Id: RECIO.tid,v 1.9 2014/07/08 12:10:15 patrick Exp $
  -- $Name:  $
  -- $Revision: 1.9 $
  -- $State: Exp $
  --
  -- $Log: RECIO.tid,v $
  -- Revision 1.9  2014/07/08 12:10:15  patrick
  -- tid added to package and install
  --
  -- Revision 1.8  2009/10/20 05:48:56  justin
  -- new path.
  --
  -- Revision 1.7  2009/01/09 07:15:51  justin
  -- this is now distributed to
  -- C:\Program Files\CodeMagusLimited\bin\
  -- hence the path change.
  --
  -- Revision 1.6  2008/11/13 05:42:22  justin
  -- removed linux path, and made codemagus\bin path.
  --
  -- Revision 1.5  2008/11/12 21:12:00  stephen
  -- Add code to allow cleanup to close open streams
  --
  -- Revision 1.4  2008/03/27 20:58:06  stephen
  -- Additions and changes for Windows
  --
  -- Revision 1.3  2008/03/27 13:16:03  stephen
  -- Fix documentation in TID file
  --
  -- Revision 1.2  2008/03/27 13:05:59  stephen
  -- Change string buffer processing and add test files
  --
  -- Revision 1.1.1.1  2008/03/27 11:41:07  stephen
  -- Initial source import
  --
}

     type : typea;
     module : "${THISTLEPORTALBINPATH}reciotai${THISTLEPORTALSUFDL}";
     init : reciotai_initial;
end.
```

2 Interface reference

2.1 Interface Methods

The following methods are exposed by the `RECIO` Type A Interface. These are shown here together with the names of their parameters and the with the associate Record Stream I/O Library function name. For a description of the method, refer to the corresponding Record Stream I/O Library documentation.

2.1.1 `error()`

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

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

Function `error()` returns the last formatted error associated with the given stream.

Parameters for Function `error()`

`stream` The stream parameter is the value returned by the `open()` function and the handle on which all Record Stream I/O operations are performed.

2.1.2 `open()`

`open()` Function `open()` is Type A Interface wrapper function for the recio library function `recio_open()`.

This function maps between the *Thistle* called function and the recio library function. If the recio 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 stream, then the function returns blob which is used a handle for operations on the stream. This blob must be passed back to all other library function in order to perform the respective operation on the stream.

Parameters for Function `open()`

`open_string` The open string supplied to the `open()` function is a text string in the format as required by the first parameter of the `recio_open()` function. This string names the access method to be used; details of the object to process; and any option settings required by the access method for the processing of the stream.

`mode` The mode parameter corresponds to the second parameter of the `recio_open()` function. The required mode is supplied using a string corresponding to the mode values:

Mode Value	String Value	Mode
RECIO_MODE_SEQ_INPUT	SEQ_INPUT	sequential input
RECIO_MODE_SEQ_OUTPUT	SEQ_OUTPUT	sequential output
RECIO_MODE_DIR_INPUT	DIR_INPUT	direct input
RECIO_MODE_DIR_OUTPUT	DIR_OUPUT	direct output
RECIO_MODE_SKIP_INPUT	SKIP_INPUT	skip sequential input

Examples

```
mode:="SKIP_INPUT"
```

`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 support:

Flag Value	Flag String	Option
RECIO_OPT_HELP	HELP	Invoked assistant for open string
RECIO_OPT_VERBOSE	VERBOSE	process with maximum message output

Examples

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

2.1.3 `close()`

`close()` Function `close()` is wrapper function for the recio function `recio_close()`.

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

Parameters for Function `close()`

`stream` The stream parameter is the value returned by the `open()` function and the handle on which all Record Stream I/O operations are performed.

2.1.4 `assist()`

`assist()` Function `assist()` wraps the `recio_assist()` function.

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

Parameters for Function `assist()`

The `assist()` function has no parameters.

2.1.5 `read()`

`read()` Function `read()` wraps the `recio_read()` function.

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

Parameters for Function `read()`

`stream` The stream parameter is the value returned by the `open()` function and the handle on which all Record Stream I/O operations are performed.

2.1.6 `write()`

`write()` Function `write()` wraps the `recio_read()` function.

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

The `write()` function does not return anything to the script.

Parameters for Function `write()`

`stream` The stream parameter is the value returned by the `open()` function and the handle on which all Record Stream I/O operations are performed.

`buffer` The buffer parameter must have a type of BLOB and is the data that will be sent to the stream.

2.1.7 `point()`

`point()` Function `point()` wraps the `recio_read()` function.

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

The `write()` function does not return anything to the script.

Parameters for Function `point()`

`stream` The stream parameter is the value returned by the `open()` function and the handle on which all Record Stream I/O operations are performed.

`buffer` The buffer parameter must have a type of BLOB and contains the key of the record in the stream to which the stream position should be changed to.

2.1.8 `tell()`

`tell()` Function `tell()` wraps the `recio_read()` function.

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

The `tell()` returns a BLOB which represents the current position in the stream. For example, for sequential input processing, this is typically the address of the last record read from the stream.

Parameters for Function `tell()`

`stream` The stream parameter is the value returned by the `open()` function and the handle on which all Record Stream I/O operations are performed.

2.1.9 `eof()`

`eof()` Function `eof()` wraps the `recio_read()` function.

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

The `eof()` returns a value of ”1” (for true) if the state of the underlying stream is at the end of the stream; otherwise the function returns the value ”0” (for false).

Parameters for Function `eof()`

`stream` The stream parameter is the value returned by the `open()` function and the handle on which all Record Stream I/O operations are performed.

2.1.10 `key2string()`

`key2string()` Function `key2string()` wraps the `recio_read()` function.

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

Function `key2string()` returns printable text string corresponding to the given key. The given key must be a BLOB and represents and internal form of the record key.

Parameters for Function `key2string()`

`stream` The stream parameter is the value returned by the `open()` function and the handle on which all Record Stream I/O operations are performed.

`key` The key parameter supplies the internal binary or BLOB format of the key in a non-display form.

2.1.11 `string2key()`

`string2key()` Function `string2key()` wraps the `recio_read()` function.

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

Function `string2key()` returns non-printable internal BLOB representing the internal form of the record key corresponding to the given text key.

Parameters for Function `string2key()`

`stream` The stream parameter is the value returned by the `open()` function and the handle on which all Record Stream I/O operations are performed.

`key` The key parameter supplies the external text or string format of the key in a display form.

2.2 Example Script

This section explains how the type A interface to `RECIO` works by examining an example script.

Assuming the following environment variable setting:

```
CODEMAGUS_AMDPATH=C:\CodeMagus\lib\%s.amd
```

This instructs the `RECIO` library to locate access method definitions in the indicated location. Access method definitions define the functionality supported by an access method; the modes and the options available and any constraints on them. The `binary` access method is a standard supplied access method used for reading and writing records using local stream files (See `binaryam.pdf` for details).

The module `reciotai` wraps the `recio` 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 RECIO;

{
  -- File: RECIO.tid
  --
  -- This is the Thistle Interface Definition for the Type A Interface to the
  -- Code Magus Limited Record Stream I/O Library. This interface makes the
  -- Record I/O Access Methods available to Thistle Scripts.
  --
  -- Copyright (c) 2008 by Code Magus Limited. All rights reserved.
  --
  -- Author: Stephen R. Donaldson [stephen@codemagus.com].
  --

  -- $Author: patrick $
  -- $Date: 2014/07/08 12:10:15 $
  -- $Id: RECIO.tid,v 1.9 2014/07/08 12:10:15 patrick Exp $
  -- $Name:  $
  -- $Revision: 1.9 $
  -- $State: Exp $
  --
  -- $Log: RECIO.tid,v $
  -- Revision 1.9  2014/07/08 12:10:15  patrick
  -- tid added to package and install
  --
  -- Revision 1.8  2009/10/20 05:48:56  justin
  -- new path.
  --
  -- Revision 1.7  2009/01/09 07:15:51  justin
  -- this is now distributed to
  -- C:\Program Files\CodeMagusLimited\bin\
  -- hence the path change.
  --
  -- Revision 1.6  2008/11/13 05:42:22  justin
  -- removed linux path, and made codemagus\bin path.
  --
  -- Revision 1.5  2008/11/12 21:12:00  stephen
  -- Add code to allow cleanup to close open streams
  --
  -- Revision 1.4  2008/03/27 20:58:06  stephen
  -- Additions and changes for Windows
  --
  -- Revision 1.3  2008/03/27 13:16:03  stephen
  -- Fix documentation in TID file
  --
  -- Revision 1.2  2008/03/27 13:05:59  stephen
  -- Change string buffer processing and add test files
  --
  -- Revision 1.1.1.1  2008/03/27 11:41:07  stephen
  -- Initial source import
  --
}

     type : typea;
     module : "${THISTLEPORTALBINPATH}reciotai${THISTLEPORTALSUFDL}";
     init : reciotai_initial;
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 recio : CodeMagusExtras.RECIO;
```

introduces `recio` 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 in its preamble, a configuration file might include:

```
CodeMagusExtras=C:\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\RECIO.tid
```

The following scripts shows how the `RECIO` interface could be used for processing record streams:

```
package RecordStreamIOSamplePackage;

{ preamble }

   created by 'Stephen Donaldson';
   description 'Demonstrate usage of RECIO Type A Interface';
   date 2007-05-10T15:28:49;
   target 'Eresia File Portal';
   interface recio : CodeMagusExtras.RECIO;

begin

   InputOpenString := "binary(C:\\temp\\in,mode=rb,recfm=f,reclen=80)";
   OutputOpenString := "binary(C:\\temp\\out,mode=wb,recfm=f,reclen=80)";

   stream := recio.Connect();

   InputStream := stream.open(open_string:=InputOpenString,
         mode:="SEQ_INPUT",flags:="/VERBOSE");
   OutputStream := stream.open(open_string:=OutputOpenString,
         mode:="SEQ_OUTPUT",flags:="/VERBOSE");

   { alternatively:

   InputStream := stream.open(InputOpenString,"SEQ_INPUT","/VERBOSE");
   OutputStream := stream.open(OutputOpenString,"SEQ_OUTPUT","/VERBOSE");

   }


   records := 0;

   repeat
      buffer := stream.read(stream:=InputStream);
      if buffer <> nil then begin
         System.WriteLn(System.Format("Input record key = %s",
            stream.key2string(stream:=InputStream,
            key:=stream.tell(stream:=InputStream))));
         stream.write(stream:=OutputStream,buffer:=buffer);
         records := records+1;
      end
   until stream.eof(stream:=InputStream);

   System.WriteLn("Number of records copied = " #
         System.Format("%08d",records));

 end.
```

Running this script produces the expected (but truncated here) output:

```
Initializing run...
CodeMagusExtras.RECIO
Interface reciotai initialised.
$Id: output.txt,v 1.1 2008/03/27 23:13:55 stephen Exp $
Input record key = 0000000000000000
Input record key = 5000000000000000
Input record key = A000000000000000
Input record key = F000000000000000
   [snip]
Input record key = 505F050000000000
Input record key = A05F050000000000
Input record key = F05F050000000000
Input record key = 4060050000000000
Input record key = 9060050000000000
Input record key = E060050000000000
Input record key = 3061050000000000
Number of records copied = 00004408
Interface reciotai cleanup complete.
$Id: output.txt,v 1.1 2008/03/27 23:13:55 stephen Exp $
Run completed...
```