source: Source Access Method Using Recio Version 1

CML00056-01

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

August 16, 2016

Contents

1 Introduction

The source access method is a module which implements the recio [1] provider interface allowing the recio [1] user interface to support reading from a source server on a remote platform.

The source access method and the source server enable the ability to read a remote recio [1] data stream in an efficient manner. This is most relevant when the remote data stream is required by a program or programs (the requester) within a system under test (SUT) where the effect of network latency needs to be minimal. Reading the data does not cause any delay or slowdown of the network traffic, as the data is received immediately it arrives thus clearing the system wide buffers within the network layer. The source access method and source server also offer the ability for a source to be shared amongst multiple requesters within the SUT, circumventing multiple streams to the same recio [1] data stream. In this instance the multiple streams are served from one source definition and each stream is processed in a round robin manner. Finally the source access method and source server allow a stream to be repeated continuously. In effect this means that the requester within the SUT never receives an end of file condition on the stream. The source server will close and re-open the stream at end of file and continue serving records from the start of the file again. This allows a test to be set up based on time limits and a constant flow of data; in other words an end of file condition would not prematurely end a defined test cycle.

Figure 1 on page 3 shows a pictorial overview of how the source access method and source server work across the network.

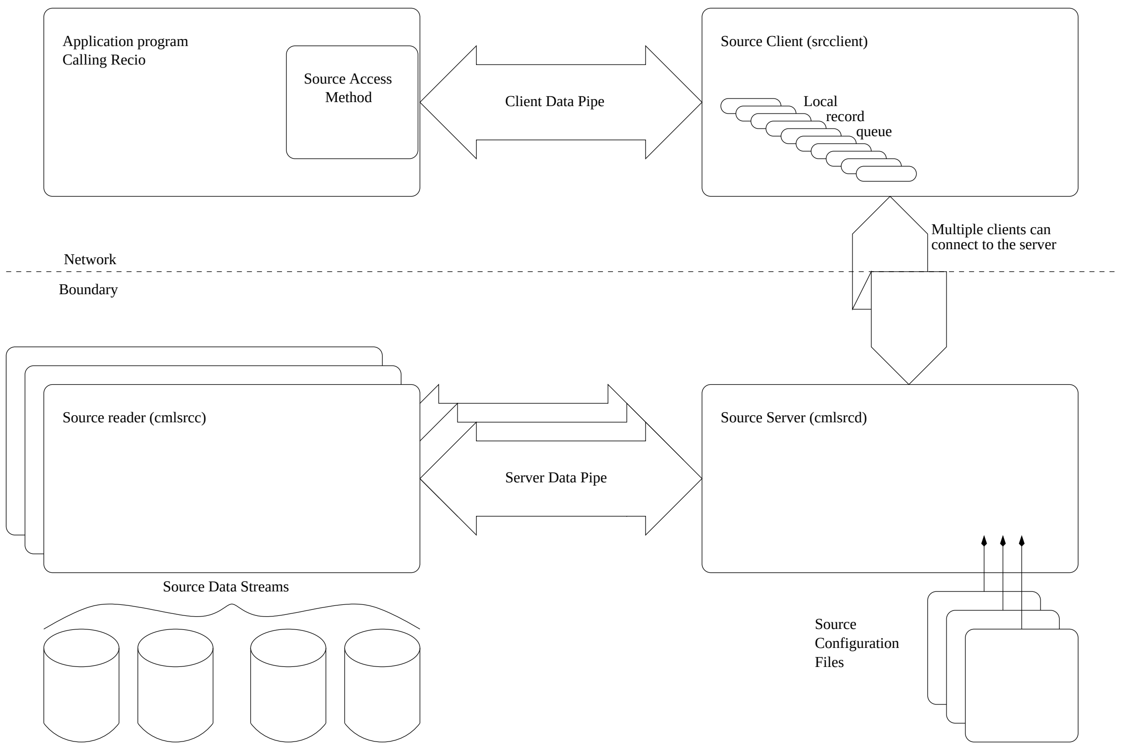


Figure 1: Overview of the Code Magus Source Access method and Server

The `source` access method starts an asynchronous process to queue records and communicate with the server. This asynchronous process uses a low and high watermark of records to control when and how many records to request from the server. Any records that it has received and not passed on to the requester are cleared from the network layer buffers and stored in the local queue. These watermark boundaries can be tuned so that optimum performance is achieved.

The `source` access method is explained in section 2 on page 4 through section 3 on page 7.

The `source` server is a daemon program that reads the source configuration file and serves records back to the `source` access method. In order to be able to process many concurrent requests, it starts an asynchronous process for each unique source request to read the `recio` [1] data stream. A unique source request is every open request for a non shared source or the first open request of a shared one.

The `source` server is explained in section 4 on page 8.

2 The `source` access method

2.1 Open String Specification

As with all `recio` [1] library open specification strings, three components comprise the open string: access method, object, and options name-value pairs.

2.2 Open Specification Access Method Name

The access method name should be specified as `source`.

2.3 Open Specification Object Name

The object name is the source name as it is know on the remote host.

2.4 Open Specification Option Name-Value Pairs

Consult the access method definition file for the option name-value pairs supported by the `source` access method. The access method definition file also supplies details of the default values (if any) of the options.

2.5 Open String Specification Examples

The following open string specification could be used during input so read from source mysource.

```
source(mysource,host=codemagus.it.nednet.co.za)
```

2.6 `source` access method definition file

The access method definition file should be consulted for the description of the options and their default values. This includes the description of the options. The access method definition file should also be consulted for the processing modes supported by the access method.

Refer the `recio` [1] library documentation for interpreting the contents of the access method definition file.

```
access source(host,port=60040);
  --
  -- File: source.amd
  --
  -- This file contains an access method definition which is used to read
  -- remote files using a common stream for multiple files.
  --
  -- Author: patrick Hayward [hayward@codemagus.com].
  --
  -- Copyright (c) 2009 Code Magus Limited. All rights reserved.
  --
  -- $Author: hayward $
  -- $Date: 2010/01/28 23:57:50 $
  -- $Id: SOURCE.amd,v 1.3 2010/01/28 23:57:50 hayward Exp $
  -- $Name:  $
  -- $Revision: 1.3 $
  -- $State: Exp $
  --
  -- $Log: SOURCE.amd,v $
  -- Revision 1.3  2010/01/28 23:57:50  hayward
  -- Fix memory creep/leeks.
  -- Change fork() to fork()..execve() so that
  -- old memory of the AM or Server that spawn
  -- children workers is overlayed by the new
  -- program. The server child is specified by
  -- a command line option and the AM child via
  -- an environment variable set in the AM or
  -- before calling the AM.
  --
  -- Revision 1.2  2010/01/27 12:20:14  hayward
  -- Remove unnused options.
  --
  -- Revision 1.1.1.1  2010/01/27 12:16:09  hayward
  -- Initial take on of sourceam code.
  --

   modes seq_input, seq_output;

   implements open;
   implements close;
   implements read;

   describe host as
      "The remote host to connect to. "
      "This may be either a host name or an IP address. "
      ;

   describe port as
      "The port on the remote host to connect to."
      ;

   -- Host can be an IP number nnn.nnn.nnn.nnn (e.g. 10.168.1.123)
```

```
    -- OR a host name (e.g. www.codemagus.com)
    constrain host as
        "^\(\(\([0-9]\{1,3\}\)\(\.[0-9]\{1,3\}\)\{3\}\)"
        "\|\(\([^. ]\+\)\(\.[^. ]\+\)*\)\)$";

    -- Port must be an integer number.
    constrain port as "^[0-9][0-9]*$";

    path = ${CODEMAGUS_AMDLIBS} "%s";
    module = "sourceam" ${CODEMAGUS_AMDSUFDL};
    entry = sourceam_init;

    -- Set the environment variable that names the client process that handles
    -- all communication to the server.
    set CODEMAGUS_SOURCE_CLIENT = ${CODEMAGUS_AMDBINS}"srcclient";

end.
```

2.7 Environment

The location and format of the access method definition file is required to be specified by the environment variable `CODEMAGUS_AMDPATH`. This environment variable supplies a pattern to the full path of where access method definition (or `amd`) files are located. The format of the environment variable is that of a path with a %s appearing in the position in which the access method member name should appear. For example, on MVS systems this might have the form:

```
CODEMAGUS_AMDPATH='DNCT00.SRDA1.AMDFILES(%s)'
```

On a Unix-based system, the value might be set in a shell profile file such as:

```
export CODEMAGUS_AMDPATH=$HOME/bin/%s.amd
```

On Windows systems, the value might be supplied from the environment variables and look something like:

```
C:\CodeMagus\bin\%s.amd
```

3 The Source Client

3.1 Introduction

The `source` access method starts a sub process that will asynchronously manage the remote access and queuing of records on its behalf. This sub process is called the `source` client. Communication between the `source` access method and the `source` client is via a two way pipe on the local machine. The `source` client connects to the `source` server and obtains from the source configuration (see section 4.3 on page 9) the low and high water marks for the record queue depth. It aims to always keep at least the low water mark and no more than the high water mark of records queued in order to supply the `source` access method on request. If the number of records queued drops below the low water mark the `source` client will request a top up from the `source` server.

3.2 Specifying the `source` client

The source client is an executable program and its fully qualified name must be specified in the environment variable `CODEMAGUS_SOURCE_CLIENT`. This may be done as per the normal way for setting an environment variable on the client platform, but is easier done by setting it in the access method definition file. An example of how it is set in the access method definition file is as follows:

For Windows

```
set CODEMAGUS_SOURCE_CLIENT = c:\CodeMagus\bin\srcclient.exe";
```

. . . or for Unix-based machines

```
set CODEMAGUS_SOURCE_CLIENT = /home/codemagus/bin/srcclient";
```

. . . or

```
set CODEMAGUS_SOURCE_CLIENT = ${HOME}"srcclient";
```

where `HOME` is another environment variable reference and makes it easy to distribute and use the same access method definition file on all machines.

4 The `source` server

4.1 Introduction

The `source` server is a program started on the server machine that waits for incoming connection requests from a `source` client, opens the requested source and serves records to the client as requested.

The source name as supplied by the `source` client is applied to a mask in order to obtain the fully qualified name of the `source` configuration file. This file once loaded specifies the attributes of the source data; the main one being a `recio` [1] open specification string that is used to open a data stream and read the data. For each unique source a `source` reader program is started that enables reading from `recio` [1] in an asynchronous manner leaving the server free to process data to and commands from the clients. All communication between the `source` server and the `source` reader is done through pipes on the machine the server is running on.

4.2 Starting and Stopping the Source Server

The Source Server program `cmlsrcd` is started from the command line and by specifying the `--help` parameter the server will print out the parameters available.

```
hayward@record:docs>cmlsrcd --help
Code Magus Limited Remote Source Server V1.0: build 2010-02-03-11.54.06
[10066]# [cmlsrcd] Id: cmlsrcd.c,v 1.7 2010/02/02 07:03:28 hayward Exp $
# Copyright (c) 2010 by Code Magus Limited. All rights reserved.
# [helpinfo@codemagus.com].
Usage: cmlsrcd [OPTION...]
  -s, --source-reader-program=<program name>     Worker program used to read
                                                 the source
  -h, --host={any|<host name>|<IP address>}      host or IP address to listen
                                                 on
  -p, --port={60040|<port number>}               Port number to accept
                                                 connections on
  -m, --source-path-mask={%s.scf|<mask>}         source configuration file mask
  -l, --log-file-mask={none|<file mask>}         Log file mask (%d=PID,
                                                 %s=program name)
  -v, --verbose                                  Verbose processing mode

Help options:
  -?, --help                                     Show this help message
  --usage                                        Display brief usage message
```

4.2.1 Source Server Start up Parameters

The start up parameters are as follows:

- `-s, --source-reader-program=<program name>`
  This required parameter specifies the `source` reader program used to read the source. It must specify a program name; either fully qualified or relative to the current directory of the running server. As distributed this program is called `cmlsrcc` on Unix based and z/OS platforms and `cmlsrcc.exe` on Windows platforms.

- `-h, --host={any|<host name>|<IP address>}`
  This optional parameter will limit the host or IP address that the server will listen on. It can be used to restrict incoming connections to only one interface. If not specified, the server will listen on all interfaces.

- `-p, --port={60040|<port number>}`
  This optional parameter specifies the port number that the server will accept connections on. If not specified, port 60040 is used.

- `-m, --source-path-mask={%s.scf|<mask>}`
  This optional parameter specifies the source configuration file mask. It should include a `%s` substitution value, which is substituted for the source name when a client connects. The default is to look in the current directory of the running server.

- `-l, --log-file-mask=<file mask>`
  This optional parameter specifies the location of the log file for any program processing output (such as verbose messages). It should include the substitution variables `%d` which is replaced with the process ID and `%s` which is replaced by the program name of the `source` server or the `source` reader. A separate log is written for each `source` reader process that runs. If not specified, the output is written under the control of `rprintf` [2].

- `-v, --verbose`
  This optional parameter will cause both the `source` server and `source` reader to produce verbose information about the processing flow of the programs.

4.3 Source Configuration File Syntax

4.3.1 Introduction

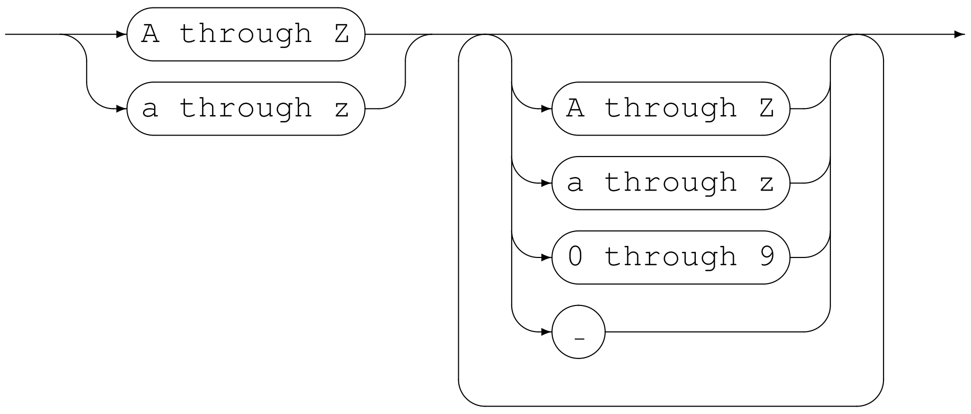
The `source` configuration file is held on the server and the name is derived from substituting the `%s` variable in the `--source-path-mask` parameter with the source name passed up from the client. This file defines all the attributes of the source data stream.

The following sections show the syntax allowed in the `source` configuration file starting with with the definitions of the basic building blocks (Identifier, String and Number) and then the actual statements. An example of a `source` configuration file can be found in appendix A on page 16.

4.3.2 *Identifier*

An *Identifier* is used as a naming value and is a made up as follows:

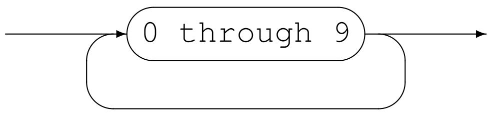
Identifier



4.3.3 *Number*

A *Number* is defined as follows:

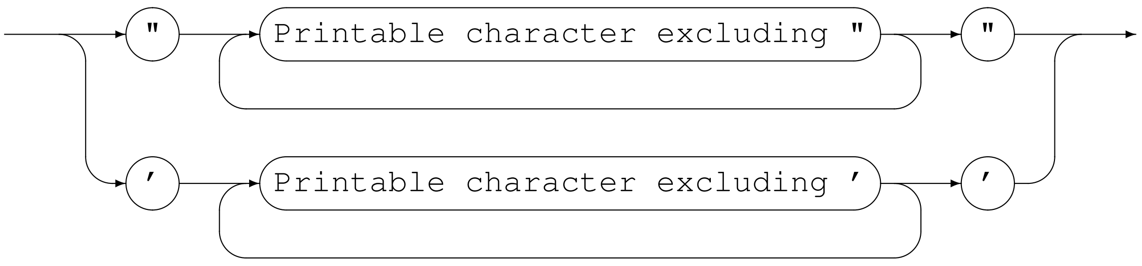
Number



4.3.4 *String*

A *String* is any number of printable characters, except the delimiter, delimited by either single or double quotes. A *String* may not span more than one line.

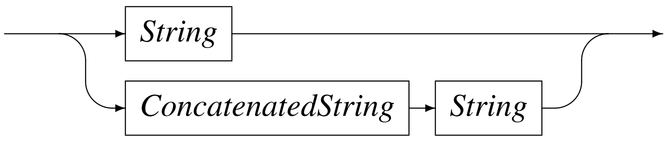
String


"
Printable character excluding "
"
'
Printable character excluding '
'

4.3.5 *ConcatenatedString*

A *ConcatenatedString* is one or more *String* values defined adjacent to each other. This is useful when defining potentially long strings that would normally span multiple lines.

ConcatenatedString



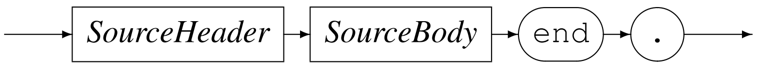
An example is:

```
"This configuration file is used "
"to read the source data residing "
"on the server"
```

4.3.6 *SourceSpecification*

The *SourceSpecification* defines the complete source configuration. It must include the header portion, the source body and be terminated be the end keyword and a full stop.

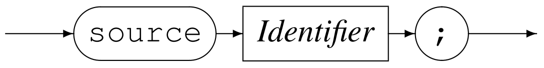
SourceSpecification



4.3.7 *SourceHeader*

The *SourceHeader* statement names the source. This name must match the name of the configuration file after the path and suffix have been removed.

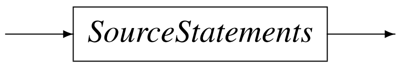
SourceHeader



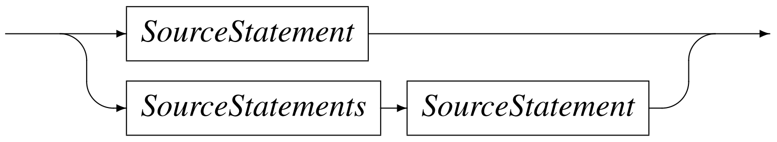
4.3.8 *SourceBody*

The *SourceBody* statements define the complete attributes of the source.

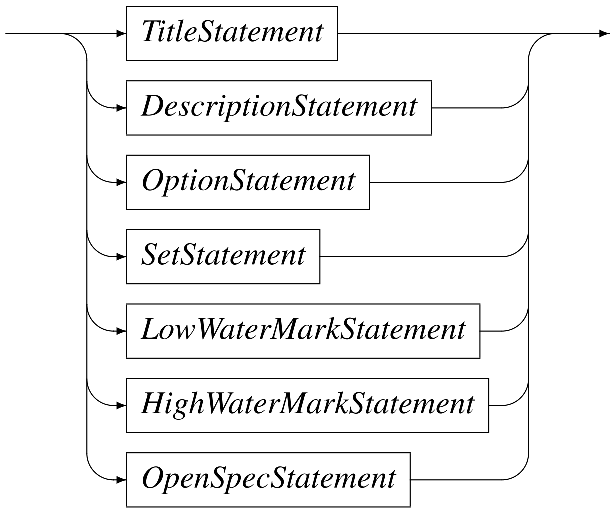
SourceBody



SourceStatements



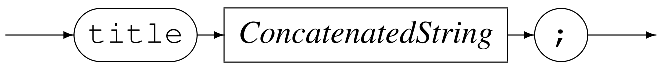
SourceStatement



4.3.9 *TitleStatement*

The *TitleStatement* defines the title of the source. It is used in any reporting of statistics for the source.

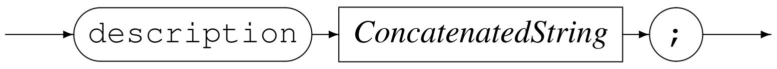
TitleStatement



4.3.10 *DescriptionStatement*

The *DescriptionStatement* describes the source in more detail than the title.

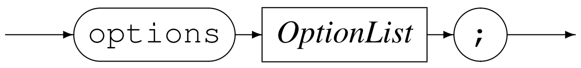
DescriptionStatement



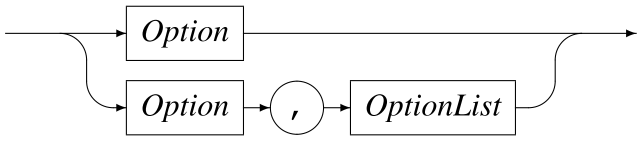
4.3.11 *OptionStatement*

The *OptionStatement* defines the options for the source.

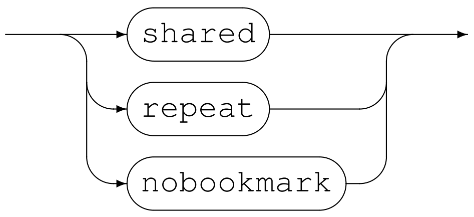
OptionStatement



OptionList



Option



Where:

- `shared`

  This means that one or more clients can connect to the same source at the same time. The server will serve records to the source in batches in a round robin manner. In other words each client will receive a portion of the file where the amount of records they receive is dependant on when they connected to the server in relation to the other clients and how quickly they consume the records and request more. Once the `source` reader program receives an end of file condition it is passed on to each client connected to that source as they ask for more records.

- `repeat`

  This means that the server will never pass an end of file condition to a client connected to the source. Instead it will close and re-open the `recio` [1] stream and continue serving records from the start of the steam again.

- `nobookmark`

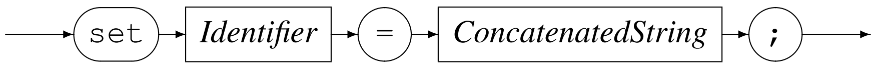
  This option cancels any bookmark statements for a shared source and forces the source to work in the same way it used to before the bookmark option was introduced; this means that once there are no clients connected to the source the source is closed. When the source is re-opened either within the same server session or after the server has been restarted reading records starts from the beginning of the stream.

Any combination or none of these options can be set for a source.

4.3.12 *SetStatement*

The *SetStatement* will set the specified environment variable to the given value on the server.

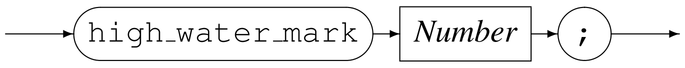
SetStatement



4.3.13 *HighWaterMarkStatement*

The high water mark defines the maximum number of records that a client can may have queued ready for delivery to the `source` access method at any one time. This value is returned to the client after it is connected.

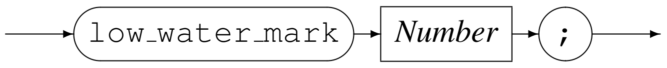
HighWaterMarkStatement



4.3.14 *LowWaterMarkStatement*

The low water mark defines the minimum number of records that the client process would like to have queued ready for delivery to the `source` access method. As soon as the number of records queued on the client falls below this value the client should request a top up back to the high water mark. This value is returned to the client after it is connected.

LowWaterMarkStatement



4.3.15 *BookMark*

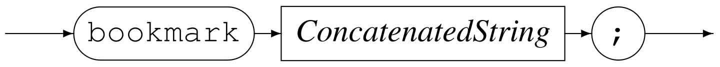
This statement is optional and if specified is only valid for sources that are shared (see option 4.3.11 on page 13). If the source is not shared (i.e. private) and this statement is specified it is ignored.

All shared sources are not closed within a single server session. When there are no more clients attached the source is kept open and is only closed once the server is stopped.

The *BookMark* statement specifies the name of a file that will be used to hold the current position of the last record within the recio data stream being read across subsequent server instances. On a server restart the recio data stream is first positioned to this point before records are returned to the client.

If a bookmark file is not specified then on a server restart the stream is opened and positioned to the beginning.

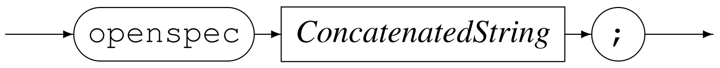
BookMarkStatement



4.3.16 *OpenSpecStatement*

The *OpenSpecStatement* defines the `recio` [1] data stream that the source will read.

OpenSpecStatement



A Sample `source` configuration file

```
-- File: SAMPLE.scf
-- This Source Configuration File (SCF) describes the source of a recio data
-- stream. A single source is able to supply data records to multiple consumers
-- on the other end of a network connection.
-- The internal source name must match the client supplied source name.
--
-- $Author: hayward $
-- $Date: 2011/10/10 16:01:46 $
-- $Id: SAMPLE.scf,v 1.6 2011/10/10 16:01:46 hayward Exp $
-- $Name:  $
-- $Revision: 1.6 $
-- $State: Exp $
--
-- $Log: SAMPLE.scf,v $
-- Revision 1.6  2011/10/10 16:01:46  hayward
-- Allow the nobookmark option in a source
-- configuration file so that bookmarking
-- can easily be turned off allowing the
-- source to revert to the original behaviour
-- (before bookmarking was added) where the
-- source is closed as soon as there are no
-- clients attached to it.
--
-- Revision 1.5  2011/09/09 10:03:14  hayward
-- Add optional source configuration statement "bookmark".
-- Only valid with shared sources (ignored for private),
-- this statement names a file that holds the current
-- position of the stream being read. The position is
-- saved in this file when the stream is closed and is
-- used to reposition the stream to this same point when
-- it is again opened in the future.
--
-- Revision 1.4  2010/02/03 16:17:16  hayward
-- Changes for documentation
--
-- Revision 1.3  2010/01/29 23:54:39  hayward
-- Remove PRIVATE as a keyword option
-- The absence of SHARED means PRIVATE.
--
-- Revision 1.2  2010/01/27 19:10:22  hayward
-- Change grammar "high water mark" and
-- "low water mark" to "high_water_mark"
-- and "low_water_mark" respectively.
--
-- Revision 1.1.1.1  2010/01/27 12:16:09  hayward
-- Initial take on of sourceam code.
--

source SAMPLE;
```

```
-- The title and description allow further explanation of the source name at
-- run time.

title "Sample Source Configuration File";
description "This is a sample source configuration and is used for "
            "Unit testing the complete source application.";

-- options:
-- repeat: when the server gets presented with the end of the sequence, it is
-- to automatically continue serving the clients, by closing the process and
-- opening the open spec again.
--
-- shared: For each client that presents itself to the server with the same
-- source name (in this case SAMPLE) will get consecutive records out of the
-- same recio stream. During the life of a server a shared source has been
-- enjanced so that it is never closed even if there are no clients attached
-- to it which effectively means that the position is always advanced in the
-- file as clients connect, retrieve and disconnect from the source. If a
-- "bookmark" file is also specified this functionality is further enhanced
-- (see the bookmark keyword further down). This enhanced functionality can
-- be disabled with the "nobookmark" keyword.
--
-- private: (not shared) For each client that presents itself to the server
-- with the same source name will get its own copy of the input stream.
--
-- nobookmark: turns off any bookmark processing. For "shared" sources only.
-- This causes the server to behave in the same way it used to before the
-- introduction of a bookmark in that the source is closed when the last
-- client using it disconnects and when it is re-opened by a new connection
-- the data stream starts from the start of the recio stream again.

   options repeat,shared;

-- Environment variables are set as they appear in the processing of the
-- config file.

set DATAHOME = ${CODEMAGUS_SOURCE}"/sourceam/testing/";

-- The low and high water marks define bounds for the number of records that
-- the client will hold in memory; if the value falls below the low water
-- mark the client will request a top up from the server. This allows the
-- client to always have records to supply to the consumer.

low_water_mark 80;
high_water_mark 100;

-- The Recio open spec defines the actual data source to read

openspec "binary("${DATAHOME}"sample.bin,mode=rb,recfm=v)";

-- The bookmark file is used to save the position of the current record
-- within the source recio stream when the stream is closed. It is only
```

```
   -- applicable to SHARED sources (ie does not affect PRIVATE sources). This
   -- allows the source to be positioned to the same place in the stream when
   -- re-opened rather than the start of the stream.
   -- This statement is optional and if it is not specified the source is opened
   -- with the RECIO mode of SEQUENTIAL INPUT and is always positioned to the
   -- start of the recio stream when opened
   -- If specified then the source is opened with the RECIO mode of SKIP
   -- SEQUENTIAL INPUT (allows positioning) and an error will occur if the Acess
   -- Method specified in the "openspec" does not support this.
   -- Wether or not this statement is specified, a shared source is never closed
   -- while the server is up and running, even if there are no clients attached
   -- to it. So during the life of the server the source always reads from the
   -- next position (and repeat will cause it to start again at EOF). It is only
   -- whan the server closes and the source is closed that the current position
   -- is recorded if a bookmark file is specified; if not specified then the
   -- source will start reading from the start of the data stream when the first
   -- client requests data from it and it is opened.
   -- Also note the stream position recorded is the position of the last read
   -- record on the server read task. As this is asynchronous to the server
   -- itself, the client sub process and the client(s), this is seldom the same
   -- position (record) that was last delivered to a client as there are many
   -- places where records are queued along the way.

   bookmark ${DATAHOME}"sample.bookmark";

end.
```

References

[1] recio: Record Stream I/O Library Version 1. CML Document CML00001-01, Code Magus Limited, July 2008. PDF.

[2] rprintf API: User Guide and Reference Version 1. CML Document CML00002-01, Code Magus Limited, July 2008. PDF.