secure: Encrypt and Compress Access Method Version 1

CML00090-01

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 `secure` access method is a module which implements the `recio` [1] provider interface allowing the `recio` user interface to support compressed and encrypted records for an underlying access method.

The default is to compress and encrypt records when reading and decrypt and decompress records when writing as is usual if the records are read or written remotely so that they are encrypted during the time spent in the network layer. Alternatively the opposite process may be performed where records are decrypted and decompressed on a read and vice versa on a write. This mode would be used, for example, if a file is read remotely and written to (and later read from) the local file system, but is required to be encrypted while there.

The encryption keys may be read from a key file. If no key file is supplied the encryption key is derived by the access method.

2 Open String Specification

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

For the `secure` access method the access method name is `secure`.

2.1 Open Specification Access Method Name

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

2.2 Open Specification Object Name

The object name should be a suitable file stream name on the local system. This is a stream file name appropriate to the local system's `fopen` file name parameter.

2.3 Open Specification Option Name-Value Pairs

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

2.4 Open String Specification Examples

In the following examples the lines have been split with a '\' in order to fit the width of the page, but should be read as one continuous string with no spaces.

The following open string specification could be used to read a variable length record binary file and compress and encrypt the record while doing so. The key `key1` is loaded from the key file example.keys in the current working directory.

```
secure(infile.rdw,using=binary,with=[recfm=v,mode=rb],\
keyfile=example.keys,keyname=key1)
```

The following open string specification could be used to write a variable length record binary file and decompress and decrypt the record while doing so. The key `key1` is loaded from the key file example.keys in the current working directory.

```
secure(outfile.rdw,using=binary,with=[recfm=v,mode=wb],\
keyfile=example.keys,keyname=key1)
```

The following example demonstrates how to read a remote file and save it in its compressed and encrypted form.

```
cmlcopy \
 -i "remote([secure(infile,using=binary,with=[recfm=v,mode=rb],\
     keysfile=/etc/keys,keyname=mykey)],\
     host=linux1,user=uname,password=pwd)" \
 -o "binary(encrypted_file.rdw,mode=wb,recfm=v)"
```

The following example shows how another application (the Code Magus file print tool) would use this file locally on the system it resides on.

```
cmlprint \
 -t object_types.objtypes
 "secure(encrypted_file.rdw,using=binary,with=[recfm=v,mode=rb],\
  onread=decrypt,keysfile=/etc/keys,keyname=mykey)"
```

Finally, this example demonstrates how to read a remote file from an MVS system in compressed and encrypted form.

```
cmlcopy \
 -i "remote([secure([DSN=MYFILE.FILE,DISP=SHR],using=mvs,\
     with=[using=binary,with=[recfm=v,mode=[rb,type=record],\
     type=record]],keysfile=DD:KEYFILE,keyname=mykey)],\
     host=mvs,user=mvsid,password=mvspwd)" \
 -o "binary(encrypted_file.rdw,mode=wb,recfm=v)"
```

3 `secure` 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 to the `recio` library documentation for interpreting the contents of the access method definition file.

```
access secure(using="binary",with="NA",onread="encrypt",onwrite="decrypt",
              keyname="NA",keyfile="NA");

  -- File: SECURE.amd
  --
  -- This file contains an access method definition which is used to read
  -- an input stream and compress and encrypt the data or to write an output
  -- stream after decrypting and expanding the data.
  --
  -- Author: Code magus Limited [www.codemagus.com].
  --
  -- Copyright (c) 2012 Code Magus Limited. All rights reserved.

  -- $Author: hayward $
  -- $Date: 2012/11/19 17:10:56 $
  -- $Id: SECURE.amd,v 1.2 2012/11/19 17:10:56 hayward Exp $
  -- $Name:  $
  -- $Revision: 1.2 $
  -- $State: Exp $
  --
  -- $Log: SECURE.amd,v $
  -- Revision 1.2  2012/11/19 17:10:56  hayward
  -- Allow opposite actions on a read or write
  -- to the default.
  -- Also allow the DES keys to be loaded from
  -- a local (to where the AM is running) file.
  --
  -- Revision 1.1.1.1  2012/11/16 15:06:45  hayward
  -- Initial import to CVS
  --

   modes seq_input, seq_output;

   implements open;
   implements close;
   implements read;
   implements write;

   describe using as
      "The option 'using' specifies the underlying access method that will "
      "be used to read the data. If specified then the object list of the "
```

```
      "Recio open string must only contain object names, otherwise the list "
      "must contain complete Recio open specifications"
      ;

   describe with as
      "The option 'with' is used only with the option 'using' and specifies "
      "the Recio options for the access method named in the 'using' option."
      ;

   describe onread as
      "The option 'onread' specifies wether to encrypt or decrypt the data "
      "as it is read."
      ;

   describe onwrite as
      "The option 'onwrite' specifies wether to encrypt or decrypt the data "
      "as it is written."
      ;

   describe keyfile as
      "The option 'keyfile' specifies a file within the local file system "
      "that holds keys generated specifically for this application. If this "
      "option is not specified it is derived from a well known source."
      ;

   describe keyname as
      "The option 'keyname' specifies the name of the key to use from the "
      "key file"
      ;

   constrain using as "^.*$";
   constrain with as "^.*$";
   constrain onread as "^\(encrypt\|ENCRYPT\|decrypt\|DECRYPT\)$";
   constrain onwrite as "^\(encrypt\|ENCRYPT\|decrypt\|DECRYPT\)$";
   constrain keyfile as "^.*$";
   constrain keyname as "^.*$";

   path = ${CODEMAGUS_AMDLIBS} "%s";
   module = "securem" ${CODEMAGUS_AMDSUFDL};
   entry = securem_init;

end.
```

4 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
```

References

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