concat: Recio Concatenate and Split Access Method Version 1

CML00067-01

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

December 15, 2020

Contents

1 Introduction

The `concat` and `split` access methods are, respectively, the input and output stream definitions of a module which implements the `recio` [1] provider interface allowing the `recio` user interface to support concatenating `recio` input streams and splitting `recio` output streams. A concatenation of input streams reads the streams in the sequence they are named in the list of objects in the `recio` open specification. When an output stream is split into more than one stream the object name is modified as each stream is opened for writing.

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.

2.1 Open Specification Access Method Name

For the `concat` access method the access method name is `concat` and for the `split` access method the access method name is `split`.

2.2 Open Specification Object Name

2.2.1 `concat` Object Name

The object name for a concatenation of input files is a comma separated list of input `recio` object names or complete `recio` open specifications. The list must be of one type or the other. The presence of the option 'with' means that the list is made up of `recio` objects and each one will be used in conjunction with the 'with' and the 'using' options to make up the open specification of each input stream. The absence of the 'with' option means that each object in the list is a complete `recio` open specification.

2.2.2 `split` Object Name

The object name list for splitting a `recio` output stream into more than one stream is formed in the same way as the `concat` object name list with one exception. Each object must have a '%s' conversion specifier embedded in it. This allows for substitution text based on the 'sequence' and 'mask' options to change the name of each file that the output stream is written to.

Usually the list of output objects is made up of only one entry; so that each output file name has a constant portion and a variable substituted portion.

However more than one output object may be specified and each will be used in turn and the last one will is treated in the same manner as for a single output object. In other words there will only be one output stream name created for each of the output objects except the last which may have more than one.

2.3 Open Specification Option Name-Value Pairs

Consult each of the access method definition files for the option name-value pairs supported by the respective access methods. The access method definition files also supply details of the default values (if any) of the options.

2.4 Open String Specification Examples

The following examples may be wrapped over multiple lines to aid readability, but should be read as a contiguous line of characters.

2.4.1 `concat` example 1

The following example will read two input files using the `text` access method.

```
concat([f1.txt,f2.txt],using=text,with=[mode=r])
```

2.4.2 `concat` example 2

The following example performs the same function as the previous one except that each object is a complete open specification.

```
concat([[text(f1.txt,mode=r)],[text(f2.txt,mode=r)]])
```

2.4.3 `concat` example 3

Of the two previous examples the first is easier to write and more intuitive especially when there are many input streams. However, the second is the only available method when the access method for each file is different as in the following example where a `text` and a `binary` file are concatenated.

```
concat([[text(f1.txt,mode=r)],[binary(f1.bin,mode=rb,recfm=v)]])
```

2.4.4 `split` example 1

The following example shows how an output stream may be split into more than one stream after each 1000 records written. The output stream in this instance writes `text` files and each file name will have the date and time it was opened as part of its name as shown below the example.

Note that the option 'sequence=time' has a granularity of 1 second. Therefore if more than 1000 records are written per second, two consecutive files will have the same name and records may be lost; as the second file opened may overwrite the first. This scenario can be avoided by opening the output stream in append mode, but would mean that some files may contain more than 1000 records.

```
split([text([f1_%s.txt],mode=r)],records=1000,sequence=time,
       mask=[D%Y%m%d_T%H%M%S])
```

Output files:

```
f1_D20100513_T172332.txt
f1_D20100513_T172333.txt
f1_D20100513_T172125.txt
f1_D20100513_T172330.txt
f1_D20100513_T172331.txt
```

2.4.5 `split` example 2

The following example shows how to split an output stream into more than one stream for every 1000 records. Each `text` output file name will have a 4 digit sequence number as part of its name as shown below the example.

```
split([text([f1_%s.txt],mode=r)],records=1000,sequence=numeric,mask=[%4.4d])
```

Output files:

```
f1_0001.txt
f1_0002.txt
f1_0003.txt
f1_0004.txt
f1_0005.txt
```

2.4.6 `split` example 3

The following example shows what happens when more than one output object is specified, in this case 2, and more than that number of output streams (`text` files in this case) are written. Again the example will split the output after each 1000 records. Each output file name will have a 4 digit sequence number as part of its name as shown below the example.

```
split([[text([f_first_%s.txt],mode=r)],[text([f_next_%s.txt],mode=r)]],
   records=1000,sequence=numeric,mask=[%4.4d])
```

Output files:

```
f_first_0001.txt
f_next_0002.txt
f_next_0003.txt
f_next_0004.txt
f_next_0005.txt
```

2.4.7 `split` example 4

The following example is the same as the previous one except that the object list is a list of objects and the 'with' and 'using' options are used to specify the underlying access method.

```
split([[f_first_%s.txt],[f_next_%s.txt]],using=text,with=[mode=w],
   records=1000,sequence=numeric,mask=[%4.4d])
```

Output files:

```
f_first_0001.txt
f_next_0002.txt
f_next_0003.txt
f_next_0004.txt
f_next_0005.txt
```

3 `concat` and `split` access method definition files

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.

3.1 `concat` AMD

```
access concat(using="NA",with="NA");

  -- File: CONCAT.amd
  --
  -- This file contains an access method definition which is used to read
  -- multiple recio streams as a concatenation of one stream. The object portion
  -- of the recio open specification is a comma separated list of either recio
```

```
  -- object names or complete open specifications.
  --
  -- Author: Code magus Limited [www.codemagus.com].
  --
  -- Copyright (c) 2010 Code Magus Limited. All rights reserved.

  -- $Author: francois $
  -- $Date: 2018/06/20 07:51:25 $
  -- $Id: CONCAT.amd,v 1.2 2018/06/20 07:51:25 francois Exp $
  -- $Name:  $
  -- $Revision: 1.2 $
  -- $State: Exp $
  --
  -- $Log: CONCAT.amd,v $
  -- Revision 1.2  2018/06/20 07:51:25  francois
  -- Hard paths removed
  --
  -- Revision 1.1  2018/06/15 10:14:07  Francois
  -- *** empty log message ***
  --
  -- Revision 1.1.1.1  2010/05/14 09:57:42  hayward
  -- Import concatam/splitam sources to CVS.
  --

   modes seq_input;

   implements open;
   implements close;
   implements read;

   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."
      ;

   constrain using as "^.*$";
   constrain with as "^.*$";

   path = ${CODEMAGUS_AMDLIBS} "%s";
   module = "concatam" ${CODEMAGUS_AMDSUFDL};
   entry = concatam_init;

end.
```

3.2 split AMD

```
access split(using="NA",with="NA",records=10,sequence="time",
   mask="D%Y%m%d_T%H%M%S");

  -- File: SPLIT.amd
  --
  -- This file contains an access method definition which allows the output of a
  -- recio stream to be written to multiple files changing the name of the
  -- output files based on size limits and a file name mask.
  -- The object in the recio string must contain a "%s" where the variable
  -- portion of the name will be inserted based on the sequence and mask.
  --
  -- Author: Code magus Limited [www.codemagus.com].
  --
  -- Copyright (c) 2010 Code Magus Limited. All rights reserved.

  -- $Author: francois $
  -- $Date: 2018/06/20 07:51:25 $
  -- $Id: SPLIT.amd,v 1.2 2018/06/20 07:51:25 francois Exp $
  -- $Name:  $
  -- $Revision: 1.2 $
  -- $State: Exp $
  --
  -- $Log: SPLIT.amd,v $
  -- Revision 1.2  2018/06/20 07:51:25  francois
  -- Hard paths removed
  --
  -- Revision 1.1  2018/06/15 10:14:07  Francois
  -- *** empty log message ***
  --
  -- Revision 1.1.1.1  2010/05/14 09:57:42  hayward
  -- Import concatam/splitam sources to CVS.
  --

   modes seq_output;

   implements open;
   implements close;
   implements write;

   describe using as
      "The option 'using' specifies the underlying access method that will "
      "be used to write 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 sequence as
      "The option 'sequence' specifies what type of sequence to replace "
      "the required '%s' with in the objects in the object list of the open "
      "specification. The 'sequence' and 'mask' must be used together and "
      "have meaningful values in order to generate distinct output file "
      "names."
      ;

   describe mask as
      "The option 'mask' specifes the mask to be used when generating the "
      "substitution text for the output file name. For a 'sequence' of 'time' "
      "or 'numeric' it must be a valid conversion specifier for the C "
      "functions strftime or fprintf respectively."
      ;

   describe records as
      "The option 'records' specifies how many records to write to an output "
      "file before closing it and opening a new file in order to continue "
      "writing the output data."
      ;

   constrain using as "^[a-zA-Z][a-zA-Z0-9]*$";
   constrain with as "^.*$";
   constrain sequence as "^\(time\|numeric\)$";
   constrain mask as "^.*$";
   constrain records as "^[0-9]\+$";

   path = ${CODEMAGUS_AMDLIBS} "%s";
   module = "concatam" ${CODEMAGUS_AMDSUFDL};
   entry = concatam_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.