dataset: Catalog Access Method Definitions Version 1

CML00013-01

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

December 15, 2020

1 Introduction

The dataset access method is a module which implements the recio [1] provider interface allowing the recio user interface to support named and cataloged configurations of access method open specification strings. These open specifications can be generic, arranged in a hierarchy, and can differ for the same cataloged name depending on the mode that the recio Record Stream is opened in. Because this named dataset object and mode resolves to an open specification string, the actual access method used is determined by the detail of the catalog entry. Hence, the dataset access method is a convenient way of abstracting away from the details of the access method open specification string.

Entries are arranged in configurations of artefect facts called catalogs. These are text files which detail the open string specifications which are located using the object of the dataset access method. Thee catalogs are arranged in a hierarchy, and the object name governs the navigation of these catalogs from a root catalog in an attempt to resolve the object name back to an open specification string.

There are a number of sections or entry types in a catalog. The first section (after an identification preamble) is optional and contains a list of set commands and values which are set at the time the catalog is loaded into storage (typically the first time the catalog is required as dictated by the search logic). The variables in these set commands refer to environment variables. The string values of these environment variables may also contain environment variable references. These are expended before the set takes place. It is an error if a string fails to expand because of the environment variable substitution.

After the optional set commands section in the catalog, a number of, but at least one, entry statements appear. There are three types of entry statement. A specific entry statement is one which will match a supplied environment dataset object name exactly, together with the supplied mode on the recio_open call. A generic entry is specified by a regular expression. In the search path, if no specific entry within a catalog matches the dataset object name, a check will be made to test the dataset object name against the regular expressions of the generic entry. The first one which matches will be used to determine, depending on the supplied mode, which open specification string, if any, will be used in the recio_open call. The third type of entry is a level entry which, if a specific or a generic entry within a catalog fails to match the object, the patterns of the level entry will be used to determine, if possible, whether a subsequent catalog should be processed in order to locate a usable entry.

When a substitution open specification string is found, the string is subjected to environment variable expansion just prior to returning the open specification string from the catalog search and the resultant open specification string will be used in the recio_open function call to open the required recio Record Stream. Additionally, the dataset object is assigned to the environment variable CATALOG_OBJECT before catalog look up, making generic catalog entries possible.

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 `dataset` access method the access method name is `dataset`.

2.1 Open Specification Access Method Name

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

2.2 Open Specification Object Name

The object name is used as the catalog search argument. This name will typically conform to a convention laid out by the site, possibly influenced by the machine types being used. For example, on an MVS machine the object name might conform to that of a `DATASET`; or on a UNIX box the name might conform to a relative or absolute path. Alternatively the names might simply follow some machine independent labelling convention.

2.3 Open Specification Option Name-Value Pairs

Consult the access method definition file for the option name-value pairs supported by the `dataset` 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 example, the object name might refer to a catalog entry that maps to an open specification string of an MVS dataset:

```
dataset(XXGG00.XXPACK.FIL010.D0.SQ1234)
```

The following similar open specification string example, overrides the default catalog root path and root catalog configuration for the resolution of the entry to the open specification string. In this example, the catalogs are expected to reside in `/home/work/testcats` and the root catalog name to use will be `TEST.cdf`.

```
dataset(XXGG00.XXPACK.FIL010.D0.SQ1234,catpath=/home/work/testcats,cata
```

The following examples illustrates a naming convention independent of any hosting machine, using a suitable convention of labels as the `dataset` objects. Note that the object name does not necessarily represent a file as the record stream could be supplied by an access method that supplies the data locally out of a call to another program or as the result set of a local or remote database query.

```
dataset(AMEX_AAA_TEST_PLASTICS)

dataset(VISA_AAA_TEST_ACCOUNTS)
```

3 `dataset` Access Method Definition

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 definition file.

```
access dataset(catpath=${CODEMAGUS_AMDCATPATH},
               catalog=${CODEMAGUS_AMDCATNAME},astype="NA");

  -- File: DATASET.amd
  --
  -- This file contains an access method definition which is used to read
  -- and write files which are indirectly referred using detrails from a
  -- catalog of open string specifications for the access method to be
  -- used. The object name in this access method together with the mode
  -- paramater are used to determin which entry in the catalog is to be
  -- used.
  --
  -- Author: Stephen R. Donaldson [stephen@codemagus.com].
  --
  -- Copyright (c) 2008 Code Magus Limited. All rights reserved.

  -- $Author: francois $
  -- $Date: 2018/06/20 07:51:27 $
  -- $Id: DATASET.amd,v 1.6 2018/06/20 07:51:27 francois Exp $
  -- $Name:  $
  -- $Revision: 1.6 $
  -- $State: Exp $
  --
  -- $Log: DATASET.amd,v $
  -- Revision 1.6  2018/06/20 07:51:27  francois
  -- Hard paths removed
  --
  -- Revision 1.1  2018/05/01 19:51:36  Francois
  -- *** empty log message ***
```

```
--
-- Revision 1.1  2018/05/01 08:44:02  Francois
-- *** empty log message ***
--
-- Revision 1.1  2018/04/06 14:34:23  Francois
-- *** empty log message ***
--
-- Revision 1.1  2018/03/18 07:12:43  Francois
-- *** empty log message ***
--
-- Revision 1.5  2014/09/04 09:33:49  hayward
-- Use "astype" instead of "alias" as the keyword
-- for an alternate name mask for catalog lookup.
--
-- Revision 1.4  2014/09/03 15:23:07  hayward
-- An name type alias can be used for the catalog lookup,
-- which helps to generate the correct open spec string for
-- a file that has a non standard name. For example
-- if a catalog entry causes all files with a suffix
-- ".*\.rdw$" (ends in .rdw) to be read with
-- "binary(${CATALOG_OBJECT},mode=r,recfm=v)"
-- and a user has a RDW file called myrdwfile.bin
-- they could still use datasetam as follows:
-- "dataset(myrdwfile.bin,astype=.rdw)"
-- OR
-- "dataset(myrdwfile.bin,astype=file.rdw)".
--
-- Revision 1.3  2008/07/30 20:36:39  stephen
-- Updates from initial testing of access method
--
-- Revision 1.2  2008/07/29 19:29:30  stephen
-- Additional dataset access method devlopment
--
-- Revision 1.1.1.1  2008/03/31 22:32:20  stephen
-- Add source to repository
--

 modes seq_input, seq_output, skip_input, dir_input, dir_output;

 implements open;
 implements close;
 implements read;
 implements write;
 implements point;
 implements tell;

 describe catpath as
    "This is the location where the root and all referenced catalogs will "
    "be found as part of the catalog search process.";

 describe catalog as
    "The catalog option is used to specify a fully qualified name of the "
```

```
      "catalog to be used to locate the correct string to used in the open.";

   describe astype as
      "The astype option specifies a different name mask, to be used instead "
      "of the object name, to look up the catalog entry.";

   constrain catpath as ".*";
   constrain catalog as ".*";
   constrain astype as ".*";

   path = ${CODEMAGUS_AMDLIBS} "%s";
   module = "datasetam" ${CODEMAGUS_AMDSUFDL};
   entry = datasetam_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
```

4.1 `dataset` Access Method Specific Environment Variables

There are a number of environment variables which are specific to the `dataset` access method:

- `CODEMAGUS_CATPATH` is used to specific the location of the catalog files if the `catpath` option is set to NULL or the environment variable `CODEMAGUS_CATENV` is set to "`YES`".

  Note that the default value of the `catpath` option may be supplied by an environment variable and that that environment variable could be difference from this

environment variable as is the case of the sample catalog `MASTCAT.cdf`.

- `CODEMAGUS_CATENV` is used to indicate that the environment variable `CODEMAGUS_CATPATH` must be used to override the `catpath` option. This is useful when testing alternative catalog configurations out of a different location from the default.

4.2 Environment Variable Expansion

There are a number of places where environment variable references are expanded during the parsing and searching of catalogs. This only applies where the environment references are expressed in the following manner, without any white spaces (expressed as a regular expression):

```
\$\{[A-Za-z][A-Za-z0-9_]*\}
```

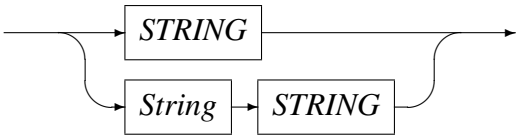
For example, "`${CATALOG_OBJECT}`".

5 Catalog Definitions

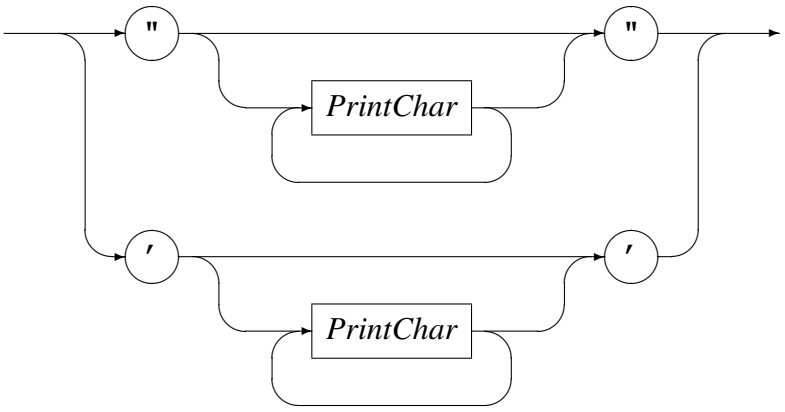
Catalogs are describes by a `catalog` configuration file. An overview of the structure of these files was presented in Section1. This section provides a reference for the language used to describe a catalog.

String and *IDENTIFIER* literals are used extensively in a catalog definition file and are explained below.

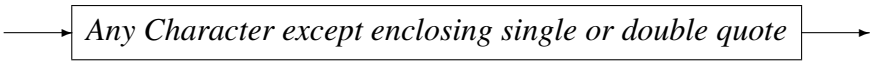
String



STRING

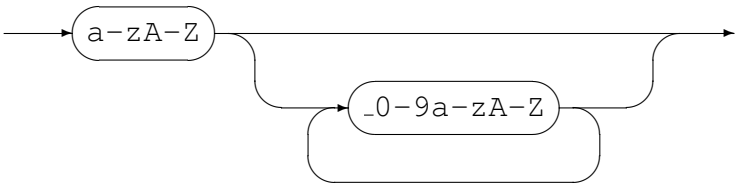


PrintChar



A String is a sequence of characters enclosed in single or double quotes, but not containing the enclosing quote. Strings may be formed by the concatenation of one or more adjacently defined strings.

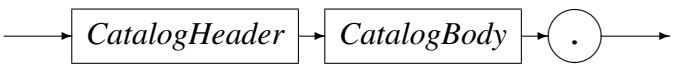
IDENTIFIER



An *IDENTIFIER* must begin with a letter of the alphabet (upper or lower case) and is optionally followed by any letter of the alphabet (upper or lower case), a digit or the under-score character.

A catalog configuration resides in a catalog definition file, hence the file name suffix of `cdf`. The catalog definition contains two parts, the *CatalogHeader* and the *CatalogBody*. The *CatalogBody* must be followed by a full-stop.

CatalogDefinition



The *CatalogHeader* identifies the file as a catalog definition file and states the name of the catalog.

CatalogHeader

The *IDENTIFIER*, and wherever *IDENTIFIER* is used, may not be a catalog definition file keyword, depicted here in a tele-type style font.

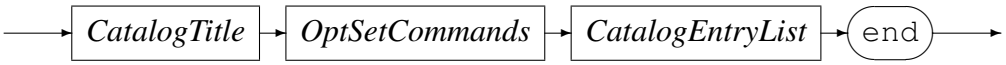
The name of the catalog in the *CatalogHeader* must match the name of the catalog definition file. For example, a catalog header of

```
catalog mastcat;
```

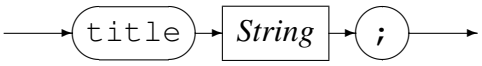
must reside in a catalog definition file called "`MASTCAT.cdf`". Note that the catalog entry name is folded to upper case and the "`.cdf`" suffix is appended and the base name must conform to an *IDENTIFIER*.

The *CatalogBody* comprises of a `title` of the catalog, optionally followed by the a section of `set` commands, and then followed by a set of catalog entries.

CatalogBody

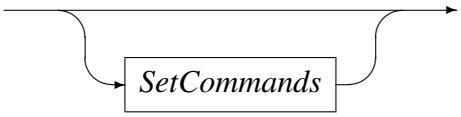


CatalogTitle

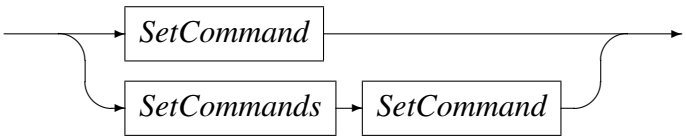


A catalog is introduced by the `title` keyword followed by one or more strings literals between quotation marks and then followed by the semi-colon. See the syntax of the *String* above.

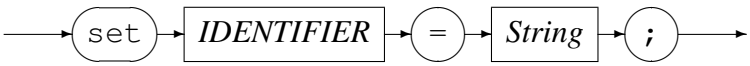
OptSetCommands



SetCommands



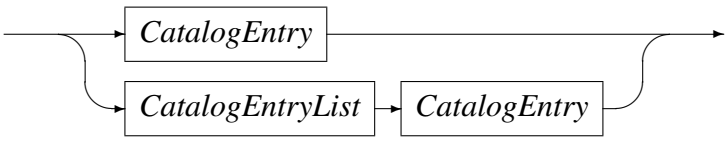
SetCommand



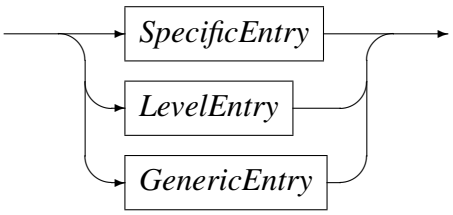
If any `set` commands are found in a catalog definition file, then they are processed as the catalog definition file is parsed. The `set` command sets the environment variable indicated by the *IDENTIFIER* to the value indicated by the supplied *String* The *String* is subjected to the environment variable expansion before being used to set the environment variable.

The *CatalogEntryList* follows any `set` commands.

CatalogEntryList

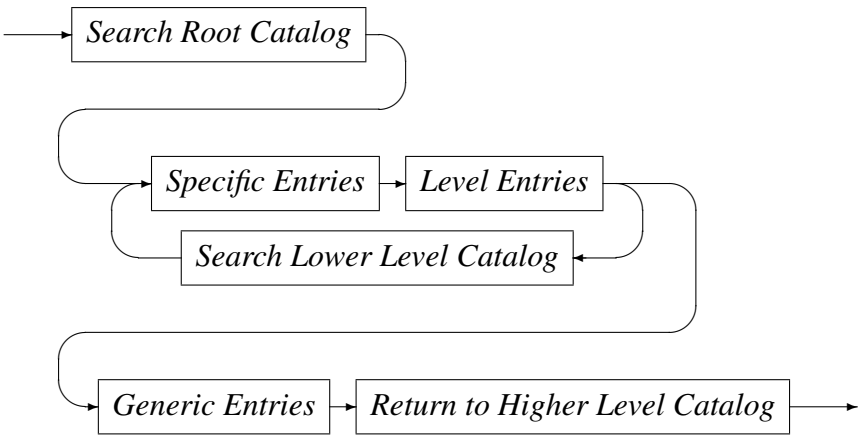


CatalogEntry



There are three different forms of *CatalogEntry* they may appear in any order within the *CatalogEntryList*, but when processed are processed by the search mechanism in the following order within a single catalog definition file.

Catalog Search Order



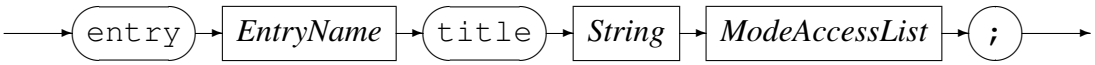
- First, given an object string from a `dataset` open specification string, the object name is checked to see whether it matches any of the specific entries defined by a *SpecificEntry* definition.
- Secondly, if the object does not match any *SpecificEntry* in the catalog definition, then the object is checked against the patterns of the *LevelEntry* definitions. If a match is made in a *LevelEntry* definition, then the search process continues in the same manner in the sub catalog referred to in the *LevelEntry* definition.
- Finally, if the object also fails to match any *LevelEntry* pattern, then the object name is checked to see if it matches any pattern of a *GenericEntry*.
- If the object does not match any entry pattern in a sub catalog the search returns to the higher level catalog, until if no match is found in the root catalog an error

is returned.

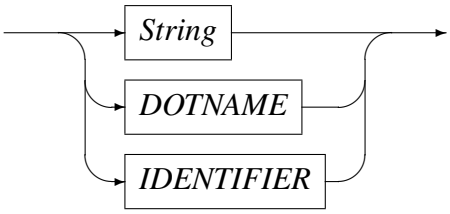
As soon as a match to a *SpecificEntry* or *GenericEntry* is found, the search is stopped and the the access method open specification string is processed.

It should be noted that if there are sub catalogs defined and they are searched due to the object matching the *LevelEntry*, all the *GenericEntry* entries of it will be processed before the *GenericEntry* entries of the higher level catalog.

Specific Entries

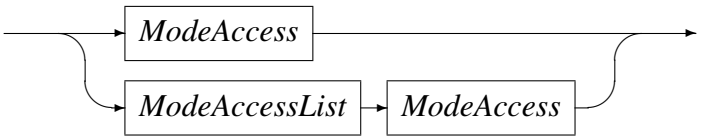


EntryName



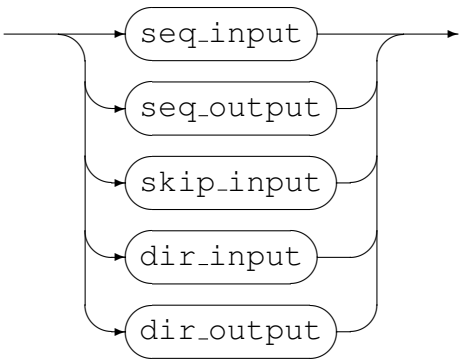
The *EntryName* in a *SpecificEntry* definition may have one of three formats. The entry may be specified as a string; as a dotted name of *IDENTIFIER*s such as an MVS style dataset name; or as a stand-alone *IDENTIFIER*. It only ever matches the object exactly.

ModeAccessList

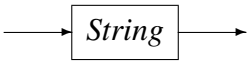


ModeAccess

OpenMode



OpenSpecString

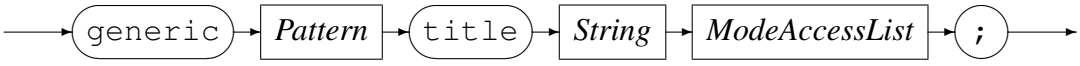


The *ModeAccessList* lists a sequence of pairs each containing a `recio` open *Mode* value together with an *OpenSpecString*. If the dataset object (or `astype` value) matches an entry then, if one of the `mode` values also matches the *mode* that the dataset is being opened with, the associated *OpenSpecString* is returned, subject to environment variable expansion, for use as the actual `recio` open string of the request.

Level Entries

The format of the *LevelEntry* is the simplest of the three types of entry. For the *LevelEntry* the *String Pattern* regular expression is expected to match the `dataset` object or `astype` value, and wherever this is the case during a catalog search, the catalog definition file indicated by the *IDENTIFIER* is used to continue the search for the object.

Generic Entries



Pattern

The format of the *GenericEntry* is similar to the *SpecificEntry* except that the *Pattern* may only be specified as a *String* and is expected to contain a regular expression to be matched to the `dataset` object or `astype` value.

6 Sample Catalog Definitions

6.1 The Master Catalog

```
catalog MASTCAT;

   title "MASTER catalog for the Recio Dataset Access Method";

-- The following environment variables are expected to be set before invoking
-- datasetam:
-- SYSB_USERID,
-- SYSB_PASSWD
--
-- The following two definitions can be overridden by setting the environment
-- variables before invoking datasetam. The default keyword will only cause
```

```
-- these environment variables to be set if they are currently not set.
   set default SYSB_HOST = "SYSB";
   set default SYSB_PORT = "60021";
   set SYSB_AUTH="user="${SYSB_USERID}",password="${SYSB_PASSWD};
   set SYSB="host="${SYSB_HOST}",port="${SYSB_PORT}","${SYSB_AUTH};

   generic "SYSAQA.*"
      title "SYSA QA generic file"
      mode seq_input
         access "remote([binary(${CATALOG_OBJECT},mode='rb,type=record')],"
                "${SYSB})"
      mode seq_output
         access "remote([binary(${CATALOG_OBJECT},mode='wb,type=record')],"
                "${SYSB})"
   ;

   entry MVSTST.CURRENT.ACCOUNT.MASTER
      title "Current Accounts Master File on MVS Test"
      mode seq_input
         access "remote([binary([${CATALOG_OBJECT}],mode='rb,type=record')],"
                "${SYSB})"
      mode seq_output
         access "remote([binary([${CATALOG_OBJECT}],mode='wb,type=record')],"
                "${SYSB})"
   ;

   generic ".*\.rdw\(log\)\?$"
      title "RDW & RDWLOG files to processed as binary variable length records"
      mode seq_input
         access
            "binary(${CATALOG_OBJECT},mode=rb,recfm=v)"

      mode seq_output
         access
            "binary(${CATALOG_OBJECT},mode=wb,recfm=v)"
   ;

   generic ".*\.txt$"
      title "Text files"
      mode seq_input
         access
            "text(${CATALOG_OBJECT},mode=r)"

      mode seq_output
         access
            "text(${CATALOG_OBJECT},mode=w)"
   ;

   generic ".*\.ccl$"
      title "Circular Log files"
      mode seq_input
         access
```

```
            "circular(${CATALOG_OBJECT})"

      mode seq_output
         access
            "circular(${CATALOG_OBJECT})"
   ;

   level "^SYSB.*$" catalog SYSB;

  -- end of MASTCAT.

end.
```

6.2 A Level Catalog

```
catalog SYSB;

   title "SYSB General Data Set Catalog";

   generic "^SYSB\.MASTER\.CLIENT\.G[0-9]\{4\}V[0-9]\{2\}$"
      title "SYSB Master client file GDG"
      mode seq_input
         access
            "remote([mvs([DISP=SHR,DSN=${CATALOG_OBJECT}],"
            "using=binary,with=[mode=[rb,type=record],recfm=f,reclen=127])],"
            "${SYSB})"

      mode seq_output
         access
            "remote([mvs([DISP=(NEW,CATLG,DELETE),DSN=${CATALOG_OBJECT},"
            "RECFM=FB,LRECL=127,BLKSIZE=27940,SPACE=(CYL,(10,10),RLSE)],"
            "using=binary,with=[mode=[wb,type=record],recfm=f,reclen=127])],"
            "${SYSB})"
   ;

   entry ALL_SYSB_TABLES
      title "List of all SYSB tables from MVS DB2"
      mode seq_input
         access "remote([db2query([SELECT * FROM SYSIBM.SYSTABLES "
                "WHERE CREATOR LIKE 'SYSB%'],SUBSYS=DSN1)],"
                "${SYSB})"
   ;


   generic ".*\.rdw\(log\)\?$"
      title "RDW & RDWLOG files to processed as binary variable length records"
      mode seq_input
         access
            "remote([binary(${CATALOG_OBJECT},mode=rb,recfm=v)],"
                "${SYSB})"
```

```
      mode seq_output
         access
            "remote([binary(${CATALOG_OBJECT},mode=wb,recfm=v)],"
                "${SYSB})"
   ;

   generic ".*\.txt$"
      title "Text files"
      mode seq_input
         access
            "remote([text(${CATALOG_OBJECT},mode=w)],"
                "${SYSB})"

      mode seq_output
         access
            "remote([text(${CATALOG_OBJECT},mode=w)],"
                "${SYSB})"
   ;

  -- end of SYSB.

end.
```

7 Example Open Specification Strings

Given the above catalogs, the following examples show *dataset* open specifications and the resulting open specification string used from the catalog search.

- Reading a file with suffix `.rdw`
  `dataset(myfile.rdw)` results in
  `binary(myfile.rdw,mode=rb,recfm=v)`
- Writing a text file
  `dataset(myfile.txt)` results in
  `text(myfile.txt,mode=w)`
- Writing a non standard name binary file
  `dataset(myfile,astype=.rdw)` results in
  `binary(myfile,mode=wb,recfm=v)`
- Writing a text file to a remote system. Here the Code Magus utility Translate file is used to write a generated file (using `testam`) of 100 records to a remote EBCDIC machine. The `astype` parameter matches the `SYSB` level entry and directs the catalog search to the SYSB sub catalog, where the generic entry "`.*\.txt$`" with mode `seq_output` is matched and returns the target open specification string. Also note that the two environment variables for the user ID and password are set before the catalog search takes place, whereas the two for the host and port

are set inside the MASTCAT catalog.

```
export SYSB_USERID=myuserid
export SYSB_PASSWD=mypwd
cmlxfile -I ascii -i "test(x,reclen=127,recfm=f,eofafter=100)" \
         -o "dataset(x.txt,astype=SYSB.txt)"
```

Results in

```
cmlxfile -I ascii -i "test(x,reclen=127,recfm=f,eofafter=100)" \
         -o "remote([text(x.txt,mode=w)],host=SYSB,port=60021,\
            user=myuserid,password=mypwd)"
```

References

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