remote: Remote Record Stream Access Method Version 1

CML00022-01

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 `remote` access method is a module which implements the Recio provider interface as a client which allows any other Recio access method to be used remotely across a `TCP/IP` network. To this extent a server is also implemented.

2 Open String Specification

As with all Recio library open specification strings, three components comprise the open string:

1. access method
2. object
3. option name-value pairs.

2.1 Open Specification Access Method Name

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

2.2 Open Specification Object Name

The `object` for a remote access request is another Recio open string specification. This is the Recio open string that will be used on the remote system. It is normally embedded in brackets '[' and ']'.

2.3 Open Specification Option Name-Value Pairs

Consult the access method definition file for the option name-value pairs supported by the `remote` access method. The access method definition file also supplies details of the default values, if any, of the options. See section 3 on page 4 for an example of an access method definition file that can be used for accessing all `remote` access method files.

Only 3 options are required, they are:

- `host`. The host name or dotted decimal IP address of the server.
- `user`. The effective user id to use when accessing data.
- `password`. The effective password of the user id.

Note that the above list does not describe all the options available.

2.4 Open String Specification Examples

In the examples below the string may be wrapped across multiple lines. This is only for convenient formatting in this document, all the strings should be seen as continuous although white space is allowed around tokens. They all use the values CMLUSER and CMLPASS for the user name and password respectively.

The following specification string can be used to read a `text` file from the host *amachine.codemagus.com*.

```
remote([text(/tmp/in,mode=r)],host=amachine.codemagus.com,
       user=CMLUSER,password=CMLPASS)
```

The following specification string can be used to write to a `binary` file on the host *amachine.codemagus.com* using the non default port 60021.

```
remote([binary(/tmp/bout,mode=wb,reclen=20,recfm=fb)],
       host=amachine.codemagus.com,port=60021,user=CMLUSER,password=CMLPASS)
```

The following specification string shows how to to access a server indirectly via another server. This may be useful when the client machine does not have direct access to the endpoint server, but knows another server that does. The client connects to *midmachine.codemagus.com* in order to read the variable binary data in `testing.dat` on machine *endpoint.codemagus.com*.

```
remote([remote([binary(testing.dat,mode=rb,recfm=v)],
       host=endpoint.codemagus.com,user=CMLUSER,password=CMLPASS)],
       host=midmachine.codemagus.com,user=CMLUSER,password=CMLPASS)
```

3 remote access method definition file

```
access remote(host,port=60020,user,password,timeout=600,batchsize="MAX");
  --
  -- File: remote.amd
  --
  -- This file contains an access method definition which is used to read
  -- and write remote files using the access method requested.
  --
  -- Author: patrick Hayward [hayward@codemagus.com].
  --
  -- Copyright (c) 2008 Code Magus Limited. All rights reserved.
  --
  -- $Author: francois $
  -- $Date: 2018/06/20 07:51:59 $
  -- $Id: REMOTE.amd,v 1.17 2018/06/20 07:51:59 francois Exp $
  -- $Name:  $
  -- $Revision: 1.17 $
  -- $State: Exp $
  --
  -- $Log: REMOTE.amd,v $
  -- Revision 1.17  2018/06/20 07:51:59  francois
  -- Hard paths removed
  --
  -- Revision 1.1  2018/05/01 19:53:49  Francois
  -- *** empty log message ***
  --
  -- Revision 1.1  2018/05/01 08:46:13  Francois
  -- *** empty log message ***
  --
  -- Revision 1.1  2018/04/06 14:36:46  Francois
  -- *** empty log message ***
  --
  -- Revision 1.1  2018/03/18 07:14:52  Francois
  -- *** empty log message ***
  --
  -- Revision 1.16  2009/07/20 18:09:13  hayward
  -- Add new functionality: batch read/write.
  -- This allows for faster sequential reads and writes
  -- by making the two peers operate asynchronously.
  -- Of course when a client or server task closes the
  -- connection prematurely (either because of an error
  -- or limiting records) there may be reords/data still
  -- in flight between the client and server that need to
  -- be mopped up in order to close the circuit neatly.
  -- The AMD option batchsize is used to determine the batch
  -- size.
  --
  -- Revision 1.15  2009/01/22 14:50:52  hayward
  -- Up the timeout to stop slow MVS systems dropping the connection.
  --
  -- Revision 1.14  2009/01/19 10:01:23  hayward
```

```
-- Improve gethostbyname() (and for Windows gethostbyaddr()) functionality.
-- Remove ADDRESSLOOKUP parameter from REMOTE.amd.
-- Use dumpbuf library to print network buffers.
--
-- Revision 1.13  2008/08/28 22:03:46  hayward
-- Changes required for Linux/Unix user name/password authentication.
--
-- Revision 1.12  2008/07/09 17:09:07  hayward
-- Changed names of some options and removed underscores.
--
-- Revision 1.11  2008/07/04 13:53:38  hayward
-- Use substitution variables for resolving DLL name.
--
-- Revision 1.10  2008/07/04 12:50:36  hayward
-- Change name of init roputine to be in line with other AM's.
--
-- Revision 1.9  2008/06/30 17:28:08  hayward
-- Remove the constraint of SECUSER and SECPASS, although both
-- user and password are still required.
--
-- Revision 1.8  2008/04/07 17:03:24  hayward
-- Make address_lookup default "NO".
--
-- Revision 1.7  2008/04/07 15:41:16  hayward
-- Add net_timeout and address_lookup as options to remote AMD.
--
-- Revision 1.6  2008/03/31 08:58:32  hayward
-- Final update for REMOTE.
--
--
 modes seq_input, seq_output, skip_input;

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

 describe host as
    "The Host to connect to for recio access. "
    "This may be either a host name or an IP address. "
    "Host is a required parameter.";

 describe port as
    "The port on the host machine to connect to. This is an optional "
    "parameter as it has a default value.";

 describe user as
    "The user is the remote system effective user ID used when reading or "
    "writing data."
    "User is a required parameter.";
```

```
   describe password as
      "The password is the remote system effective password associated "
      "with the user ID used when reading or writing data."
      "Password is a required parameter.";

   describe timeout as
      "timeout is the maximum time in seconds that the client will wait "
      "to receive a message before giving up and closing the connection. "
      "A value of zero means that the client will not time out.";

   describe batchsize as
      "batchsize allows better throughput for sequential access "
      "by only synchronising the client and server after this many "
      "records have been transmitted.";

   -- 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\}\)"
       "\|\(\([^. ]\+\)\(\.[^. ]\+\)*\)\)$";

   -- User id must be at least one character. I.e. it must be supplied.
   constrain user as "^..*$";

   -- Password must be at least one character. I.e. it must be supplied.
   constrain password as "^..*$";

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

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

   -- batchsize must be an integer or MAX.
   constrain batchsize as "^\(MAX\|[0-9]\+\)$";

   path = ${CODEMAGUS_AMDLIBS} "%s";
   module = "remoteam" ${CODEMAGUS_AMDSUFDL};
   entry = remoteam_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 `UNIX` (and `MVS` under a Unix File System) based system, the value might be set in a shell profile file such as:

```
export CODEMAGUS_AMDPATH=/usr/local/CodeMagus/bin/%s.amd
```

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

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

5 Server Administration

5.1 UNIX and Windows

5.1.1 Installation

Unpack the `Remote` package in a suitable directory (typically `/usr/local/CodeMagus`) with the following command:

```
tar -xvzf package_file.tgz
```

where `package_file.tgz` is the name of the package received from Code Magus.

5.1.2 Starting the server

The server is started by running the program `rmtserv`. By default it will listen for connections on any interface on the default port. There are specific environment variables that must be set; the script `rmtserv.sh` is an example of how to start the server after setting all the environment variables. The parameters to `rmtserv` are as follows:

```
rmtserv --help
[rmtserv] $Id: remoteam.tex,v 1.19 2011/11/23 11:41:10 hayward Exp $
Build: Jul 20 2009 at 19:01:40
Copyright (c) 2008 by Code Magus Limited. All rights reserved.
[www.codemagus.com].

Code Magus Limited Remote Recio Server Version 1.: build 2009-07-20-19.01.39
Usage: rmtserv [OPTION...]
  -h, --host={ANY|<hostname|IP address>}     IP address to listen on.
  -p, --port={60020|<Port Number>}           IP port to listen on.
  -c, --child=<program name>                 Name of child program.
  -U, --user=<user name>                     Non secure user name for client
                                             programs.
  -P, --password=<password>                  Non secure password for client
                                             programs.
  -v, --verbose                              Verbose printing during
                                             processing.

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

- `host`. This parameter can be specified as a host name or IP address and if specified, the server will only accept connections on that the IP address. If not specified the server will accept connections on any interface defined to the local machine.
- `port`. If specified determines the number of the port the server will listen on. The default is 60020.

- child. This specifies the fully qualified name of the sub-task program. If this is not specified then the default sub-task program rmtchild is invoked from the current working directory of the running server.
- user. This specifies a default user name that a client can use when connecting to the server. If a client process connects with this user name and the correct password the session is considered to be authenticated and all file access is done under the security context of the server. Be aware that this could be a security hole and should be switched off in a data sensitive system especially if the server is started with super user privileges. This is often used when running the server on a system for which there is no current hook into the user authentication services or on a development system. If this parameter is not specified then user authentication is always performed against the underlying security system and all file access is done only under a correctly authenticated user.
- password. This specifies the default password associated with the default user. A client process must supply both the correct user and password (which are case sensitive) before being considered authentic.

5.1.3 Stopping the server

The server can be stopped by either a KILL (cancel on MVS) or an INTERRUPT signal. An INTERRUPT close allows currently executing sub-tasks to complete.

5.2 MVS

5.2.1 Overview

On MVS the Remote Recio Server is installed in the same way as it is for UNIX. The server and sub task programs must reside and be executed out of the Unix File System. This means that the server can be run within z/OS UNIX System Services as per the UNIX instructions or as is more usual on MVS the JCL procedure SRDRMTSV can be initiated as a started task. Templates of all JCL and environment variable configuration files can be found in the Remote package and are shown in the Appendix.

5.2.2 Setting up an Unix File System

It is recommended that a separate Unix File System is created and mounted within z/OS UNIX System Services. Creating and mounting the Unix File System will usually need to be done by a system administrator using the following example commands:

1. Create the Unix File System in MVS. This is preferably a ZFS or otherwise an HFS See appendix D on page 18 or appendix E on page 19 respectively for an example JCL deck.
2. Create a mount point in z/OS UNIX System Services.

   ```
   mkdir -p /usr/local/CodeMagus/
   ```

3. Mount the Unix File System.

   For example to mount a ZFS

   ```
   /usr/sbin/mount -t zfs -f SYS2.CODE.MAGUS.ZFS /usr/local/CodeMagus/
   ```

   Or to mount an HFS

   ```
   /usr/sbin/mount -t hfs -f SYS2.CODE.MAGUS.HFS /usr/local/CodeMagus/
   ```

   To have this Unix File System mounted permanently an entry can be added to the BPXPRMxx member in the system parmlib. An example is as follows:

   ```
   MOUNT    FILESYSTEM('SYS2.CODE.MAGUS.ZFS')
            TYPE(ZFS)
            MODE(RDWR)
            MOUNTPOINT('/usr/local/CodeMagus')
   ```

4. Unpack the Remote package in the new Unix File System with the following commands:

   ```
   cd /usr/local/CodeMagus/
   pax -r -f package_file.tgz
   ```

   where package_file.tgz is the name of the package received from Code Magus.

After unpacking the Remote package the Unix File System should have the following files and directory structure:

```
/Mountpoint
+-- bin
|   |
|   +-- srdrmtsv
|   +-- srdrmtch
|   +-- REMOTE.amd
|   +-- MVS JCL decks
|
+-- lib
|   |
|   +-- remoteam.so
|
`-- logs
```

5. Change the attributes of srdrmtsv and srdrmtch so that they are marked as program controlled. This allows the server to switch security context to the user making the remote request from the user running the RemoteServer. Use the following command in the bin directory shown above:

```
extattr +p srdrmtsv
extattr +p srdrmtch
```

6. Copy the JCL and Environment files to a PDSE.

   Examples of these files are shown in appendix A on page 15 and onward. Copy and customise the following files:

   - srdrmtsv.proc - The main procedure for running the RemoteServer as a started task. See section 5.2.3 on page 11.
   - srdrmtsv.jcl - Allows the RemoteServer to be run as a standard job.
   - srdrmtsv.env - This file holds definitions of all environment variables required by the RemoteServer. See section 5.2.5 on page 12.
   - srdzfscr.jcl - Sample JCL to create the ZFS.
   - srdhfscr.jcl - Sample JCL to create the HFS.

5.2.3 Customising JCL Start up

The file srdrmtsv.proc should be customised and placed in a JCL procedure library from which it can be started with the MVS START command or from within a job using an EXEC statement. An example of the procedure is in appendix A on page 15. The following changes are required in order to start the server from the console without having to retype the parameters each time:

- Change the parameter HFSPATH to match the mount point chosen previously.

- Change the parameter HLQ to match the high level qualifier of the MVS data sets that will hold the environment variable configuration file.
- (optional) Change the value of the parameter ENVIRON if required to conform to local system standards.
- (optional) Change the location and attributes of the log files for stderr and stdout by changing any of the parameters LOGPATH, STDOUT or STDERR.
- (optional) Change the value of the parameter DIAG. This can be set to '-v' in order to obtain extra diagnostics when running the server.

If the server is to be run via a submitted job then it is preferable to edit the jobs parameters. See appendix B on page 17 for an example.

5.2.4 RACF Considerations

If the server is to be run as a started task then changes are required to be made in the system security server. The following points relate to RACF, but can be used as a guide for any other third party security server.

The following commands show example RACF commands detailing how to allow a procedure to be started on the system.

```
SETROPTS GENERIC(STARTED)
RDEFINE STARTED <jobname>.* STDATA(USER(<user>) GROUP(<group>) TRUSTED(YES))
SETROPTS CLASSACT(STARTED) RACLIST(STARTED)
```

Where :

1. <jobname> is the started task job name. This is SRDRMTSV in the example procedure in appendix A on page 15.
2. <user> is the user name chosen for the started task to run under.
3. <group> is the name of the RACF group profile

5.2.5 Environment variable Set Up

The file srdrmtsv.env which is referred to as DD-name STDENV in the procedure srdrmtsv.proc must be customised for the current system. An example of the file can be found in appendix C on page 17. The file must be saved in the PDSE dataset referred to as STDENV in the procedure. If this PDSE does not exist it should be created. This allows an installation to continue using the same environment variable configuration file when a new Remote package is distributed.

The following changes are required:

- Change HFSPATH to match the mount point chosen previously. Typically this is /usr/local/CodeMagus.

5.2.6 Starting the Remote Server

The server can be started by either:

- issuing the start command from the MVS console; an example is:

  ```
  START SRDRMTSV,HFSPATH='/u/cml',ENVIRON=CMLVARS,HLQ=CML,DIAG='-v'
  ```

  Note the lower case parameters; one for the path name and one for the verbose parameter. If the procedure srdrmtsv.proc is correctly configured then the command becomes simply:

  ```
  START SRDRMTSV
  ```

- submitting the JCL job srdrmtsv.jcl shown in appendix B on page 17. Configure the job to refer to the correct procedure library that holds the procedure srdrmtsv.proc.

Once running the job can be monitored in MVS as normal or in z/OS UNIX System Services with the command:

```
ps -ef | grep srdrmtsv
```

Where srdrmtsv is the server program name.

5.2.7 Stopping the MVS Remote Server

The server can be stopped with an MVS cancel command. This will immediately terminate any file access currently in progress. A better way to close the server is to use the kill command in z/OS UNIX System Services. Find the PID of the server with the ps command and use the following to close it:

```
kill -2 1234
```

where 1234 corresponds to the PID of the server.

5.2.8 Running the Remote Server from z/OS UNIX System Services

The remote server can also be run directly in z/OS UNIX System Services in exactly the same way it is for any other UNIX system. Refer to section 5 on page 8 for more information.

5.2.9 Using other access method modules

The Remote package only delivers the functionality to access data remotely. To actually read the data (via the remote access method) other access methods are required to be installed. Each individual access method is deployed into and runs out of the Unix File System as well. An access method is usually packaged with an access method definition file, an executable program and a User manual. They should be unpacked using a similar command to that used for the Remote package in the same place as the remote access method and RemoteServer.

NOTE: All access methods must have the extended attribute of 'program controlled' in order to work under the RemoteServer. An example of the command to do this for the binary access method is:

```
extattr +p /usr/local/CodeMagus/lib/binary.so
```

All the environment variables are specified in the STDENV file (see section 5.2.5 on page 12). The following list, although not exhaustive, lists the main access methods required.

1. BINARY. Access to variable and fixed length binary data.
2. TEXT. Access to newline delimited text based data.
3. DIR. Access to the MVS catalog using the MVS Catalog search Interface (CSI). This interface accepts meta characters which allows the use of a mask to match dataset names. The meta characters are the same as those used in ISPF option 3.4.
4. MVS. Access to any other access methods after first allocating an MVS dataset using JCL syntax.
5. IMAGE. Access to a DB2 image copy dataset.
6. REMOTE. Access to another remote recio server.

Each access method package should also include a User Guide; which holds information specific to that access method.

A MVS Server JCL procedure

```
//SRDRMTSV PROC HLQ=HAYWARD.A,ENVIRON=SRDRMTSV,                          X
//         HFSPATH='/usr/local/CodeMagus',                               X
//         DIAG=,                                  '-v' for verbose      X
//         LOGPATH='/tmp',                                               X
//         STDOUT=SRDRMTSV.&SYSNAME..OUT.TXT,                            X
//         STDERR=SRDRMTSV.&SYSNAME..ERR.TXT
//*
//* FILE: srdrmtsv.prc
//* This PROC will run the remote server srdrmtsv.
//* Associated environment variables can be set in the dataset
//* named by DD:STDENV.
//*
//*
//* $Author: hayward $
//* $Date: 2008/11/05 13:00:05 $
//* $Id: srdrmtsv.proc,v 1.6 2008/11/05 13:00:05 hayward Exp $
//* $Name:  $
//* $Revision: 1.6 $
//* $State: Exp $
//*
//* $Log: srdrmtsv.proc,v $
//* Revision 1.6  2008/11/05 13:00:05  hayward
//* Updates with respect to running the server and Access Methods
//* from the HFS under USS.
//*
//* Revision 1.5  2008/06/16 18:11:02  hayward
//* Changes to PROC parameters. Allows SYSNAME as part of output
//* logs for a started task - useful on a multi-host system.
//*
//* Revision 1.4  2008/06/16 14:58:12  hayward
//* Correct parameter name HFSPATH.
//*
//* Revision 1.3  2008/06/16 13:44:52  hayward
//* Chnages to PROC and JCL for using BPXBATCH.
//*
//* Revision 1.2  2008/06/11 09:59:19  hayward
//* Complete install and admin instructions for MVS.
//*
//* Revision 1.1  2008/06/10 22:02:28  hayward
//* Finalise changes to Remote server jobs.
//*
//RMTSERV   EXEC PGM=BPXBATCH,TIME=1440,                                 X
// PARM='PGM &HFSPATH./bin/srdrmtsv &DIAG -c &HFSPATH./bin/srdrmtch'
//STEPLIB  DD DSN=&HLQ..LOADLIB,DISP=SHR
//STDOUT   DD  PATH='&LOGPATH./&STDOUT',                                 X
//         PATHOPTS=(OWRONLY,OCREAT,OTRUNC),                             X
//         PATHMODE=(SIRWXU,SIRWXG,SIRWXO)
//STDERR   DD  PATH='&LOGPATH./&STDERR',                                 X
//         PATHOPTS=(OWRONLY,OCREAT,OTRUNC),                             X
//         PATHMODE=(SIRWXU,SIRWXG,SIRWXO)
```

```
//STDENV   DD  DSN=&HLQ..ENVIRON(&ENVIRON),DISP=SHR
//SYSPRINT DD  SYSOUT=*
//SYSOUT   DD  SYSOUT=*
//SYSUDUMP DD  SYSOUT=*
//*
//          PEND
```

B MVS Server JCL job

```
//SRDRMTSV JOB  (@@@@),'CML',CLASS=A,MSGCLASS=X,NOTIFY=&SYSUID,
// REGION=0M
//*
//      JCLLIB ORDER=(HAYWARD.A.PROCLIB)
//*
//      SET SYSNAME=ZBOX
//*
//* File: rmtserv.jcl
//* This JCL deck will run rmtserv.sh under BPXBATCH.
//* It can be used to set the required environment variables for
//* the shell script.
//*
//* $Author: hayward $
//* $Date: 2008/11/05 13:00:05 $
//* $Id: srdrmtsv.jcl,v 1.3 2008/11/05 13:00:05 hayward Exp $
//* $Name:  $
//* $Revision: 1.3 $
//* $State: Exp $
//*
//* $Log: srdrmtsv.jcl,v $
//* Revision 1.3  2008/11/05 13:00:05  hayward
//* Updates with respect to running the server and Access Methods
//* from the HFS under USS.
//*
//* Revision 1.2  2008/06/16 13:44:52  hayward
//* Chnages to PROC and JCL for using BPXBATCH.
//*
//* Revision 1.1  2008/06/10 22:02:28  hayward
//* Finalise changes to Remote server jobs.
//*
//SRDRMTSV EXEC SRDRMTSV HLQ=HAYWARD.A,DIAG='-vV'
//*
//
```

C MVS Server Environment Variable configuration file

```
CODEMAGUS_AMDPATH=HFSPATH/bin/%s.amd
CODEMAGUS_AMDLIBS=HFSPATH/lib/
CODEMAGUS_AMDSUFDL=.so
CODEMAGUS_AMDCATPATH=HFSPATH/bin/
CODEMAGUS_AMDCATNAME=MASTCAT
_EDC_ADD_ERRNO2=1
CODEMAGUS_OUTPUT_PREFIX_TS=1
```

D Sample JCL for creating a ZFS

```
//CMLZFS   JOB  (@@@@),'CML',CLASS=A,MSGCLASS=X,NOTIFY=&SYSUID
//*
//* File: srdzfscr.jcl
//* This JCL deck will create the specified ZFS file.
//*
//* $Author: hayward $
//* $Date: 2009/10/19 11:15:11 $
//* $Id: srdzfscr.jcl,v 1.1 2009/10/19 11:15:11 hayward Exp $
//* $Name:  $
//* $Revision: 1.1 $
//* $State: Exp $
//*
//* $Log: srdzfscr.jcl,v $
//* Revision 1.1  2009/10/19 11:15:11  hayward
//* Add ZFS create template to CVS.
//*
//DEFINE   EXEC PGM=IDCAMS
//SYSPRINT DD  SYSOUT=*
//SYSUDUMP DD  SYSOUT=*
//AMSDUMP  DD  SYSOUT=*
//SYSIN    DD     *
    DELETE SYS2.CODE.MAGUS.ZFS CL PURGE
    SET MAXCC = 0
    DEFINE CLUSTER (NAME(SYS2.CODE.MAGUS.ZFS) -
           VOLUMES(vvvvvv) -
           LINEAR CYL(500 50) SHAREOPTIONS(2))
/*
//*
//CREATE   EXEC PGM=IOEAGFMT,REGION=0M,
// PARM=('-aggregate SYS2.CODE.MAGUS.ZFS -compat')
//SYSPRINT DD  SYSOUT=*
//STDOUT   DD  SYSOUT=*
//STDERR   DD  SYSOUT=*
//SYSUDUMP DD  SYSOUT=*
//CEEDUMP  DD  SYSOUT=*
//*
//
```

E Sample JCL for creating an HFS

```
//SRDHFSCR JOB  (@@@@),'CML',CLASS=A,MSGCLASS=X,NOTIFY=&SYSUID
//*
//* File: srdhfscr.jcl
//* This JCL deck will create the specified HFS file.
//*
//* $Author: hayward $
//* $Date: 2009/10/19 11:36:27 $
//* $Id: srdhfscr.jcl,v 1.3 2009/10/19 11:36:27 hayward Exp $
//* $Name:  $
//* $Revision: 1.3 $
//* $State: Exp $
//*
//* $Log: srdhfscr.jcl,v $
//* Revision 1.3  2009/10/19 11:36:27  hayward
//* Change HLQ2 value.
//*
//* Revision 1.2  2008/11/05 13:00:05  hayward
//* Updates with respect to running the server and Access Methods
//* from the HFS under USS.
//*
//* Revision 1.1  2008/06/16 13:44:52  hayward
//* Chnages to PROC and JCL for using BPXBATCH.
//*
//*
//      SET HLQ=SYS2
//      SET HLQ2=CODE.MAGUS
//*
//DELETE   EXEC PGM=IEFBR14
//RMTHFS   DD  DSN=&HLQ..&HLQ2..HFS,                                  X
//         DISP=(MOD,DELETE,DELETE),                                   X
//         SPACE=(TRK,0)
//*
//SRDHFSCR EXEC PGM=IEFBR14
//RMTHFS   DD  DSN=&HLQ..&HLQ2..HFS,                                  X
//         DISP=(NEW,CATLG,DELETE),                                    X
//         SPACE=(CYL,(50,10)),                                        X
//         VOL=SER=vvvvvv,                                             X
//         RECFM=U,                                                    X
//         DSORG=PO,                                                   X
//         DSNTYPE=HFS
//
```