Department of Information Systems
and Operations Management

BSF4ooRexx 850 Beta: Exploiting ooRexx 5.0 Direct and Redirecting Command Environment Handlers Implemented in Java/NetRexx (... and ooRexx! ;))

Making it *really* easy to create fully fledged Rexx command handlers

International RexxLA Symposium, 2022-09 (https://www.RexxLA.org)

Prof. Rony G. Flatscher

Overview

- ooRexx 5.0
  - About Rexx exit and Rexx command handlers
  - New direct and redirectable command handlers
- BSF4ooRexx 850 beta
  - Brief history of the BSF4ooRexx Rexx handler support in BSF4ooRexx 641
  - Redevising and reimplementing the Rexx handler support
    - Exploiting the new ooRexx 5.0 native APIs for Rexx handlers
    - Overview of the Java exit and command handler classes
  - Configuring Rexx instances and Rexx exit and Rexx command handlers
    - Samples
- Roundup

REXX Exit and REXX Command Handlers

- C APIs (cf. ooRexx' rexxapi.pdf documentation)
  - Callbacks to C functions for specific execution stages of a Rexx program
    - "Rexx exit handler", e.g.
      - RXOFNC called at the beginning of the search of an external function or routine
        - Handler can take over and handle the call
      - RXCMD called when a command is about to be processed
        - Handler can take over and process the command itself
      - RXVALUE called when VALUE() gets invoked with an unknown environment
  - Callbacks to C functions to handle commands from Rexx programs
    - "Rexx command handlers", e.g.
      - Writing a command handler to ease interfacing with software environments
        - E.g. allowing Rexx to issue commands to an editor or any software for which a Rexx handler got written
        - Quite popular and exploited in the mainframe environment, but very interesting for other environments as well (→ "JDOR" command handler presentation)

ooRexx and REXX Handlers

- C APIs (cf. ooRexx' rexxapi.pdf documentation, continued)
  - ooRexx allows configuring Rexx instances (RI) with handlers
  - Any Rexx program being run by such a RI can exploit the Rexx handlers
- BSF4ooRexx
  - Since 2012 samples, cf. samples/Java/handlers and samples/NetRexx/handlers
- ooRexx 5.0
  - Introduced redirectable Rexx command handlers employing and extending the optional ANSI Rexx "ADDRESS ... WITH" keyword statement
    - Configuration of a Rexx instance (RI) supports the redirectable command handlers
    - Added a new API to allow native (C++) programs to add Rexx command handlers at runtime in addition to the preconfigured ones

BSF4ooRexx 641 and REXX Handlers

- Introduced full support for REXX handlers in 2012
  - REXX exit and REXX command handlers can be implemented in Java and NetRexx
    - Abstract class org.rexxla.bsf.engines.rexx.RexxHandler
      - Defines static methods to support the Java REXX handlers like *raiseCondition(...)*, *checkCondition(...)*, *getLocalEnvironment(...)*, *setContextVariable(...)*, *getContextVariable(...)*, *dropContextVariable(...)*, *setContextVariableToNil(...)*,and a few more
    - Interface class org.rexxla.bsf.engines.rexx.RexxExitHandler
      - Defines the constants matching the REXX exit handler constants for C++
      - A Java exit handler needs to implement the following interface method

        public int handleExit(Object slot, int exitNumber, int subFunction, Object[] parmBlock)
    - Interface class org.rexxla.bsf.engines.rexx.RexxCommandHandler
      - A Java command handler needs to implement the following interface method

        public Object handleCommand(Object slot, String address, String command)

BSF4ooRexx850 and REXX Handlers, 1

- Java baseline is Java 8
  - Default interface methods become available and get exploited!
- Abstract class org.rexxla.bsf.engines.rexx.RexxHandler becomes an interface class
    - All static methods get reimplemented as default interface methods
      - Existing handlers need to remove the class qualifier "`RexxHandler.`" from the method invocations such that the default instance methods can be found instead
    - Added new methods (*newArray(...)*, *newDirectory(...)*, *newStem(...)*, *newStringTable(...)*
- RexxExitHandler extends RexxHandler thereby inheriting all its default methods
- RexxCommandHandler extends RexxHandler thereby inheriting all its features
- A new class RexxRedirectingCommandHandler extends RexxCommandHandler
  - Adding default methods for all new ooRexx redirecting APIs (cf. rexxapi.pdf)
- Cf. Java documentation: `BSF4ooRexx850 → Information → docs.bsf4oorexx → index.html`

BSF4ooRexx850 and REXX Handlers, 2

- New BSF external Rexx function BSFCommandHandler() with the following two functionalities
  - *call BsfCommandHandler 'add', environmentName, javaRexxHandler*
    - Adds dynamically a Java implemented Rexx (direct or redirecting) command handler that serves the command environment named environmentName
    - From that moment on this command handler takes over handling of commands sent with the ADDRESS keyword statement to environmentName
  - *array=BsfCommandHandler('list')*
    - Returns an array of strings denoting the currently registered Java command handlers and the environment name each handler serves
    - Indicates whether the Java command handler is a direct or a redirecting one

Prolog: A RXCMD Exit Handler, 1

- Exit handler gets invoked when a command handler is about to be called
- Can take over processing or leave it to the command handler
- Cf. rexxapi.pdf, "1.15.2.4. RXCMD" (ooRexx 5.0 documentation)
- Example
  - Will not let delete commands through to the operating system
    - If the delete command is "rm" an error condition gets raised in the Rexx program
    - If the delete command is "del" a failure condition gets raised in the Rexx program
  - The Java exit command handler will receive a "slot" argument
    - An opaque argument that needs to be passed on, if invoking default handler methods
    - Works on the current Rexx program's context, hence *must not* be cached!

Prolog: A RXCMD Exit Handler, 2

- The Java exit handler implementation

```
// cf. rexxapi.pdf, "1.15.2.4. RXCMD", this handler will not forward "del" or "rm"
// commands to the command handler, rather it raises an error or a failure condition
public int handleExit(Object slot, int exitNumber, int subFunction, Object[] parmBlock)
{
    boolean flag[]=(boolean[]) parmBlock[0]; // fetch flag array (to raise failure or error conditions)
    String address=(String) parmBlock[1];
    String command=(String) parmBlock[3];

    String [] arrCommand = command.split("\\p{javaWhitespace}+");
         if (arrCommand[0].toUpperCase().equals("DEL")) flag[0] = true; // raise failure
    else if (arrCommand[0].equals("rm"))                flag[1] = true; // raise error

    if (flag[0] || flag[1]) return   RXEXIT_HANDLED;    // do not process command
    return RXEXIT_NOT_HANDLED;  // let command handler process command
}
```

Prolog: A RXCMD Exit Handler, 2b

- C++ definitions for RXCMD in rexx.h and rexxref.pdf (rexxapidefs.h)

Represented as an array of boolean in Java

```
/***    Subfunction RXCMDHST -- Process Host Commands     */

typedef  struct _RXCMD_FLAGS {          /* fl */
   unsigned rxfcfail : 1;              /* Command failed.            */
   unsigned rxfcerr  : 1;              /* Command ERROR occurred.    */
}  RXCMD_FLAGS;

typedef  struct _RXCMDHST_PARM {        /* rx */
   RXCMD_FLAGS       rxcmd_flags;      /* error/failure flags        */
   CONSTANT_STRING   rxcmd_address;    /* Pointer to address name.   */
   unsigned short    rxcmd_addressl;   /* Length of address name.    */
   CONSTANT_STRING   rxcmd_dll;        /* dll name for command.      */
   unsigned short    rxcmd_dll_len;    /* Length of dll name.        */
   CONSTANT_RXSTRING rxcmd_command;    /* The command string.        */
   RXSTRING          rxcmd_retc;       /* Pointer to return buffer   */
}  RXCMDHST_PARM;
```

parmBlock argument

Prolog: A RXCMD Exit Handler, 3

- The Java main method that configures a Rexx instance (RI)

```
public static void main (String args[]) throws BSFException
{
    if (args.length==0) // no command line arguments supplied
    {
        System.err.println("usage:\n\tjava JavaRunRexxMonitorCommands someRexxProgram.rex");
        System.exit(-1);
    }
    String programName = args[0];           // get Rexx program name
    BSFManager mgr       =new BSFManager(); // create an instance of BSFManager
    RexxEngine rexxEngine=(RexxEngine) mgr.loadScriptingEngine("rexx");  // get Rexx
    // Configure the RexxEngine, add this class' system exit to monitor commands
    RexxConfiguration rexxConf=rexxEngine.getRexxConfiguration();
    System.err.println("RexxConf=<"+rexxConf+">");
    rexxConf.addExitHandler(RXCMD, new JavaRunRexxMonitorCommands() );
    System.err.println("RexxConf=<"+rexxConf+">");
    // Rexx code to run, just call supplied programName
    String rexxCode= "call '"+programName+"'  ;" ;
    // invoke the interpreter and run the Rexx program
    try                 { rexxEngine.apply (programName, 0, 0, rexxCode, null, null); }
    catch (Throwable t) {System.err.println(t);}    // show error, if any
    mgr.terminate();    // make sure that the Rexx interpreter instance gets terminated!
    System.exit(0);     // exit Java
}
```

Prolog: A RXCMD Exit Handler, 4

- Rexx program testMonitor.rex trying to issue delete commands

```
address system "echo hi"  -- o.k.
say "command ["sourceLine(.line-1)"]"
say "   returned: rc=["rc"] state: .rs=[".rs"]";say

address system ""         -- o.k.
say "command ["sourceLine(.line-1)"]"
say "   returned: rc=["rc"] state: .rs=[".rs"]";say

address system 'rm'       -- gets intercepted and raises an error condition
say "command ["sourceLine(.line-1)"]"
say "   returned: rc=["rc"] state: .rs=[".rs"]";say

address system 'del *.*'  -- gets intercepted and raises a failure condition (severe)
say "command ["sourceLine(.line-1)"]"
say "   returned: rc=["rc"] state: .rs=[".rs"]"
```

Prolog: A RXCMD Exit Handler, 5

- Output of running the Rexx program with the Java exit handler

```
E:\exit_handler>java JavaRunRexxMonitorCommands testMonitor.rex
RexxConf=<org.rexxla.bsf.engines.rexx.RexxConfiguration[initialAddressEnvironment=[null],externalCallPath=[
null],externalCallExtensions=[.rxj,.rxo,.rxjo,.jrexx],loadRequiredLibrary={},exitHandlers={},commandHandler
s={},redirectingCommandHandlers={}]>
RexxConf=<org.rexxla.bsf.engines.rexx.RexxConfiguration[initialAddressEnvironment=[null],externalCallPath=[
null],externalCallExtensions=[.rxj,.rxo,.rxjo,.jrexx],loadRequiredLibrary={},exitHandlers={RXCMD/3/JavaRunR
exxMonitorCommands@42110406},commandHandlers={},redirectingCommandHandlers={}]>
hi
command [address system "echo hi"  -- o.k.]
   returned: rc=[0] state: .rs=[0]

command [address system ""         -- o.k.]
   returned: rc=[0] state: .rs=[0]

command [address system 'rm'       -- gets intercepted and raises an error condition]
   returned: rc=[] state: .rs=[1]

    13 *-* address system 'del *.*'  -- gets intercepted and raises a failure condition (severe)
       >>>   "del *.*"
command [address system 'del *.*'  -- gets intercepted and raises a failure condition (severe)]
   returned: rc=[] state: .rs=[-1]
```

ooRexx 5.0 Direct Command Handler, 1

- Direct REXX command handler
- Can be dynamically loaded using BsfCommandHandler()
- Cannot handle redirected input, output and/or error
- Defines default methods named
  - *isRedirectable()* returns `false`
  - *toString(slot, environmentName)* returns a String describing this instance

ooRexx 5.0 Direct Command Handler, 2

- Java command handler named OneCommandHandler

```
import org.rexxla.bsf.engines.rexx.*;
public class OneCommandHandler implements RexxCommandHandler
{
    public Object handleCommand(Object slot, String address, String command)
    {
        System.err.println("[OneCommandHandler] address=["+address+"] "+"command=["+command+"]");
        String [] arrCommand = command.split("\\p{javaWhitespace}+"); // split at (and consume) whitespace
        String cmd = "";
        if (arrCommand.length>0)
            cmd=arrCommand[0].toUpperCase();    // get command, uppercase it
        switch (cmd)
        {
            case "HELLO":   // return values set the RC variable in Rexx
                return "Hello, ooRexx! This is your Java OneCommandHandler greeting you!";
            case "INFO":    // return information about this handler
                return toString(slot,address);
            default:        // unknown command received create an error
                String [] strAdditional = new String[]{ "Unknown command ["+command+"]" };
                //                                                  RC:
                raiseCondition(slot, "Error", command, strAdditional, "-1");
                return null;    // uses fourth argument of raiseCondition as RC
        }
    }
}
```

ooRexx 5.0 Direct Command Handler, 3

- Rexx program using the Java OneCommandHandler

```
call BsfCommandHandler 'add', 'ONE', .bsf~new("OneCommandHandler")
say "(Rexx) list of Java command handlers:"
do line over bsfCommandHandler('list')
   say "--->" line
end
address one
say "default environment got changed to" pp(address())

"hello"
say "answer to command" pp(sourceline(.line-1)) "was:" pp(rc) ".rs="pp(.rs)

"this is an unknown command"  -- this will raise an error condition
say "answer to command" pp(sourceline(.line-1)) "was:" pp(rc) ".rs="pp(.rs)

address one info
say "answer to command" pp(sourceline(.line-1)) "was:" pp(rc) ".rs="pp(.rs)
info
say "answer to command" pp(sourceline(.line-1)) "was:" pp(rc) ".rs="pp(.rs)

::requires "BSF.CLS"    -- get ooRexx-Java bridge
```

ooRexx 5.0 Direct Command Handler, 4

- Output of running the Rexx program

```
(Rexx) list of Java command handlers:
---> name: ONE             redirectable: 0 handler: OneCommandHandler@3581c5f3
default environment got changed to [ONE]
[OneCommandHandler] address=[ONE] command=[hello]
answer to command ["hello"] was: [Hello, ooRexx! This is your Java OneCommandHandler greeting you!] .rs=[0]
[OneCommandHandler] address=[ONE] command=[this is an unknown command]
answer to command ["this is an unknown command"  -- this will raise an error condition] was: [-1] .rs=[1]
[OneCommandHandler] address=[ONE] command=[INFO]
answer to command [address one info] was: [OneCommandHandler[environmentName=ONE,isRedirectable=false]] .rs=[0]
[OneCommandHandler] address=[ONE] command=[INFO]
answer to command [info] was: [OneCommandHandler[environmentName=ONE,isRedirectable=false]] .rs=[0]
```

ooRexx 5.0 Redirecting Command Handler, 1

- Redirecting REXX command handler
  - Input, output and error can be redirected to the handler
  - Can be streams but also ooRexx collection objects
  - Defines default methods to make the new APIs available (cf. rexxapi.pdf)
- Can be dynamically loaded using BsfCommandHandler()
- Defines default methods named
  - *isRedirectable()* returns `true`
  - *toString(slot, environmentName)* returns a String describing this instance
  - *isRedirectionRequested(slot)*, *isInputRedirected(slot)*, *isOutputRedirected(slot)*, *isErrorRedirected(slot)*, *areOutputAndErrorSameTarget(slot)*, *readInput(slot)*, *readInputBuffer(slot)*, *writeOutput(slot,String)*, *writeOutputBuffer(slot,String)*, *writeError(slot,String)*, *writeErrorBuffer(slot,String)*

ooRexx 5.0 Redirecting Command Handler, 2

- Java command handler named DeuxCommandHandler

```
import org.rexxla.bsf.engines.rexx.*;
public class DeuxCommandHandler implements RexxRedirectingCommandHandler
{
    public Object handleCommand(Object slot, String address, String command)
    {
        if (isOutputRedirected(slot))   // output redirected?
            writeOutput(slot, "[DeuxCommandHandler] address=["+address+"] "+"command=["+command+"]");
        else
            System.err.println("[DeuxCommandHandler] address=["+address+"] "+"command=["+command+"]");
        String [] arrCommand = command.split("\\p{javaWhitespace}+"); // split at (and consume) whitespace
        String cmd = "";
        if (arrCommand.length>0) cmd=arrCommand[0].toUpperCase();     // get command, uppercase it
        switch (cmd)
        {
            case "HELLO":   // return values set the RC variable in Rexx
                return "Hello, ooRexx! This is your Java DeuxCommandHandler greeting you!";
            case "INFO":    // return information about this handler
                return toString(slot,address);
            default:        // unknown command received create a failure
                String [] strAdditional = new String[]{ "Unknown command ["+command+"]" };
                if (isErrorRedirected(slot))    // error redirected? If so supply error/failure information
                    writeError(slot, "["+address+"]: "+strAdditional[0]);
                //                                                  RC:
                raiseCondition(slot, "Failure", command, strAdditional, "-2");
                return null;    // uses fourth argument of raiseCondition as RC
        }
    }
}
```

ooRexx 5.0 Redirecting Command Handler, 3a

- Rexx program testRedirectedPlain.rex

```
call BsfCommandHandler 'add', 'ONE',  .bsf~new("OneCommandHandler")
call BsfCommandHandler 'add', 'DEUX', .bsf~new("DeuxCommandHandler")
say "(Rexx) list of Java command handlers:"
do line over bsfCommandHandler('list')
   say "--->" line
end
address deux
say "default environment got changed to" pp(address())

"hello"
say "answer to command" pp(sourceline(.line-1)) "was:" pp(rc) ".rs="pp(.rs)

"this is an unknown command"  -- this will raise a failure (!) condition
say "answer to command" pp(sourceline(.line-1)) "was:" pp(rc) ".rs="pp(.rs)

address one info
say "answer to command" pp(sourceline(.line-1)) "was:" pp(rc) ".rs="pp(.rs)
info
say "answer to command" pp(sourceline(.line-1)) "was:" pp(rc) ".rs="pp(.rs)

::requires "BSF.CLS"    -- get ooRexx-Java bridge
```


Prof. Rony G. Flatscher

ooRexx 5.0 Redirecting Command Handler, 3b

- Output of rexx testRedirectedPlain.rex

```
(Rexx) list of Java command handlers:
---> name: DEUX            redirectable: 1 handler: DeuxCommandHandler@73c6c3b2
---> name: ONE             redirectable: 0 handler: OneCommandHandler@3581c5f3
default environment got changed to [DEUX]
[DeuxCommandHandler] address=[DEUX] command=[hello]
answer to command ["hello"] was: [Hello, ooRexx! This is your Java DeuxCommandHandler greeting you!] .rs=[0]
[DeuxCommandHandler] address=[DEUX] command=[this is an unknown command]
    13 *-* "this is an unknown command"  -- this will raise a failure condition
       >>>   "this is an unknown command"
       +++   "RC(-2)"
answer to command ["this is an unknown command"  -- this will raise a failure condition] was: [-2] .rs=[-1]
[OneCommandHandler] address=[ONE] command=[INFO]
answer to command [address one info] was: [OneCommandHandler[environmentName=ONE,isRedirectable=false]] .rs=[0]
[DeuxCommandHandler] address=[DEUX] command=[INFO]
answer to command [info] was:
[DeuxCommandHandler[environmentName=DEUX,isRedirectable=true,isRedirectionRequested=false,isInputRedirected=fal
se,isOutputRedirected=false,isErrorRedirected=false,areOutputAndErrorSameTarget=false]] .rs=[0]
```

ooRexx 5.0 Redirecting Command Handler, 4a

- Rexx program testRedirected.rex (redirecting output and error)

```
call BsfCommandHandler 'add', 'ONE',  .bsf~new("OneCommandHandler")
call BsfCommandHandler 'add', 'DEUX', .bsf~new("DeuxCommandHandler")
say "(Rexx) list of Java command handlers:"
do line over bsfCommandHandler('list')
   say "--->" line
end
arrOut=.array~new
arrErr=.array~new
address deux with output append using (arrOut) error append using (arrErr)
say "default environment got changed to" pp(address())

"hello"
say "answer to command" pp(sourceline(.line-1)) "was:" pp(rc) ".rs="pp(.rs)

"this is an unknown command"  -- this will raise a failure condition
say "answer to command" pp(sourceline(.line-1)) "was:" pp(rc) ".rs="pp(.rs)

address one info
say "answer to command" pp(sourceline(.line-1)) "was:" pp(rc) ".rs="pp(.rs)
info
say "answer to command" pp(sourceline(.line-1)) "was:" pp(rc) ".rs="pp(.rs)
say "-"~copies(79)
say "arrOut:"; say arrOut
say "-"~copies(79)
say "arrErr:"; say arrErr

::requires "BSF.CLS"    -- get ooRexx-Java bridge
```

ooRexx 5.0 Redirecting Command Handler, 4b

- Output of rexx testRedirected.rex (redirecting output and error)

```
(Rexx) list of Java command handlers:
---> name: DEUX            redirectable: 1 handler: DeuxCommandHandler@73c6c3b2
---> name: ONE             redirectable: 0 handler: OneCommandHandler@3581c5f3
default environment got changed to [DEUX]
answer to command ["hello"] was: [Hello, ooRexx! This is your Java DeuxCommandHandler greeting you!] .rs=[0]
    15 *-* "this is an unknown command"  -- this will raise a failure condition
       >>>   "this is an unknown command"
       +++   "RC(-2)"
answer to command ["this is an unknown command"  -- this will raise a failure condition] was: [-2] .rs=[-1]
[OneCommandHandler] address=[ONE] command=[INFO]
answer to command [address one info] was: [OneCommandHandler[environmentName=ONE,isRedirectable=false]] .rs=[0]
answer to command [info] was:
[DeuxCommandHandler[environmentName=DEUX,isRedirectable=true,isRedirectionRequested=true,isInputRedirected=fals
e,isOutputRedirected=true,isErrorRedirected=true,areOutputAndErrorSameTarget=false]] .rs=[0]
--------------------------------------------------------------------------------
arrOut:
[DeuxCommandHandler] address=[DEUX] command=[hello]
[DeuxCommandHandler] address=[DEUX] command=[this is an unknown command]
[DeuxCommandHandler] address=[DEUX] command=[INFO]
--------------------------------------------------------------------------------
arrErr:
[DEUX]: Unknown command [this is an unknown command]
```


Prof. Rony G. Flatscher

ooRexx 5.0 New Direct Command Handler sample/Java/handlers/command

- Direct REXX command handler
- Can be dynamically loaded using BsfCommandHandler()
- Cannot handle redirected input, output and/or error
- Defines default methods named
  - *isRedirectable()* returns `false`
  - *toString(slot, environmentName)* returns a String describing this instance
- The class implementing the command handler must be public

ooRexx 5.0 Preconfiguring Rexx, 1

- Java program RunRexxProgram
  - Preconfigures a direct Java REXX command handler to serve an environment named ONE
    - Uses the public class OneCommandHandler
  - Preconfigures a direct Java REXX command handler to serve an environment named DEUX
    - Uses the public class DeuxCommandHandler
  - Preconfigures the default environment to be ONE
  - Runs the Rexx program that got supplied as a command line argument

ooRexx 5.0 Preconfiguring Rexx, 2

- Java program named RunRexxProgram

```
import org.apache.bsf.BSFManager;
import org.rexxla.bsf.engines.rexx.*;
public class RunRexxProgram
{
    public static void main (String args[])
    {
        try
        {
            BSFManager mgr= new BSFManager();
            RexxEngine re = (RexxEngine) mgr.loadScriptingEngine("rexx");
            RexxConfiguration rexxconf=re.getRexxConfiguration();
            System.err.println("default rexxconf=["+rexxconf+"]\n");
            rexxconf.addCommandHandler("ONE"    ,         new OneCommandHandler());
            rexxconf.addCommandHandler("DEUX"   ,         new DeuxCommandHandler());
            rexxconf.setInitialAddressEnvironment("ONE");
            System.err.println("edited  rexxconf=["+rexxconf+"]\n");
            String rexxCode= "call \""+args[0]+"\"      ;" +    // Rexx code to run
                             "::requires BSF.CLS        ;" ;    // get ooRexx support (camouflage Java as ooRexx)
            re.apply (args[0], 0, 0, rexxCode, null, null);
            mgr.terminate();    // make sure that the Rexx interpreter instance gets terminated
        }
        catch (Throwable t)
        {
            System.err.println(t);
        }
        System.exit(0);     // exit Java
    }
}
```

ooRexx 5.0 Preconfiguring Rexx, 3a

- Rexx program testPreconfigured.rex

```
say "(Rexx) list of Java command handlers:"
do line over bsfCommandHandler('list')
   say "--->" line
end
say
say "current (preconfigured) environment:" pp(address())
say
"hello"
say "answer to command" pp(sourceline(.line-1)) "was:" pp(rc) ".rs="pp(.rs)
address deux "hello"
say "answer to command" pp(sourceline(.line-1)) "was:" pp(rc) ".rs="pp(.rs)
say
info
say "answer to command" pp(sourceline(.line-1)) "was:" pp(rc) ".rs="pp(.rs)
address deux info
say "answer to command" pp(sourceline(.line-1)) "was:" pp(rc) ".rs="pp(.rs)

::requires "BSF.CLS"    -- get ooRexx-Java bridge
```


Prof. Rony G. Flatscher

ooRexx 5.0 Preconfiguring Rexx, 3b

- Output of rexx testPreconfigured.rex

```
Default rexxconf=[org.rexxla.bsf.engines.rexx.RexxConfiguration[initialAddressEnvironment=[null],externalCallPath=[null],
externalCallExtensions=[.rxj,.rxo,.rxjo,.jrexx],loadRequiredLibrary={},exitHandlers={},commandHandlers={},redirectingCommandHandlers={}]]

Edited rexxconf=[org.rexxla.bsf.engines.rexx.RexxConfiguration[initialAddressEnvironment=[ONE],externalCallPath=[null],
externalCallExtensions=[.rxj,.rxo,.rxjo,.jrexx],loadRequiredLibrary={},exitHandlers={},commandHandlers={ONE=OneCommandHandler@1d44bcfa},redirectingCommandHandlers={DEUX=DeuxCommandHandler@266474c2}]]

(Rexx) list of Java command handlers:
---> name: DEUX            redirectable: 1 handler: DeuxCommandHandler@266474c2
---> name: ONE             redirectable: 0 handler: OneCommandHandler@1d44bcfa

current (preconfigured) environment: [ONE]

[OneCommandHandler] address=[ONE] command=[hello]
answer to command ["hello"] was: [Hello, ooRexx! This is your Java OneCommandHandler greeting you!] .rs=[0]
[DeuxCommandHandler] address=[DEUX] command=[hello]
answer to command [address deux "hello"] was: [Hello, ooRexx! This is your Java DeuxCommandHandler greeting you!] .rs=[0]

[OneCommandHandler] address=[ONE] command=[INFO]
answer to command [info] was: [OneCommandHandler[environmentName=ONE,isRedirectable=false]] .rs=[0]
[DeuxCommandHandler] address=[DEUX] command=[INFO]
answer to command [address deux info] was:
[DeuxCommandHandler[environmentName=DEUX,isRedirectable=true,isRedirectionRequested=false,isInputRedirected=false,isOutputRedirected=false,isErrorRedirected=false,areOutputAndErrorSameTarget=false]] .rs=[0]
```


Prof. Rony G. Flatscher

Direct and Redirecting Command Handler

- REXX exit and REXX command handlers can be implemented in ooRexx
  - Use the external BSF function *BsfCreateRexxProxy()*
    - Third argument needs to be the name of one of the following abstract classes
      - org.rexxla.bsf.engines.rexx.AbstractExitHandler
      - org.rexxla.bsf.engines.rexx.AbstractDirectCommandHandler
      - org.rexxla.bsf.engines.rexx.AbstractRedirectingCommandHandler
  - Check out the samples
    - Exit handler implemented in ooRexx only

      samples/Java/handlers/exitHandlers/04_RXMSQ/rexxonly
    - Command handlers implemented in ooRexx only

      samples/Java/handlers/commandHandlers/30_java_starter850

      samples/NetRexx/handlers/commandHandlers/30_java_starter850

Direct and Redirecting Command Handler

- Roundup
  - Supports the new ooRexx 5.0 direct and redirecting command handlers
    - In addition to C++ now Java and NetRexx can be used to implement command handlers to ease interaction with complex software systems for Rexx programmers
      - It is even possible to implement command handlers directly and only in ooRexx
    - Will be part of the planned BSF4ooRexx850 beta
    - The new external BSF function named *BsfCommandHandler()* allows for
      - Adding Java implemented command handlers at runtime of a Rexx program
      - Returning an array of currently active Java implemented command handlers with their type
    - Development is finished
- Questions ?