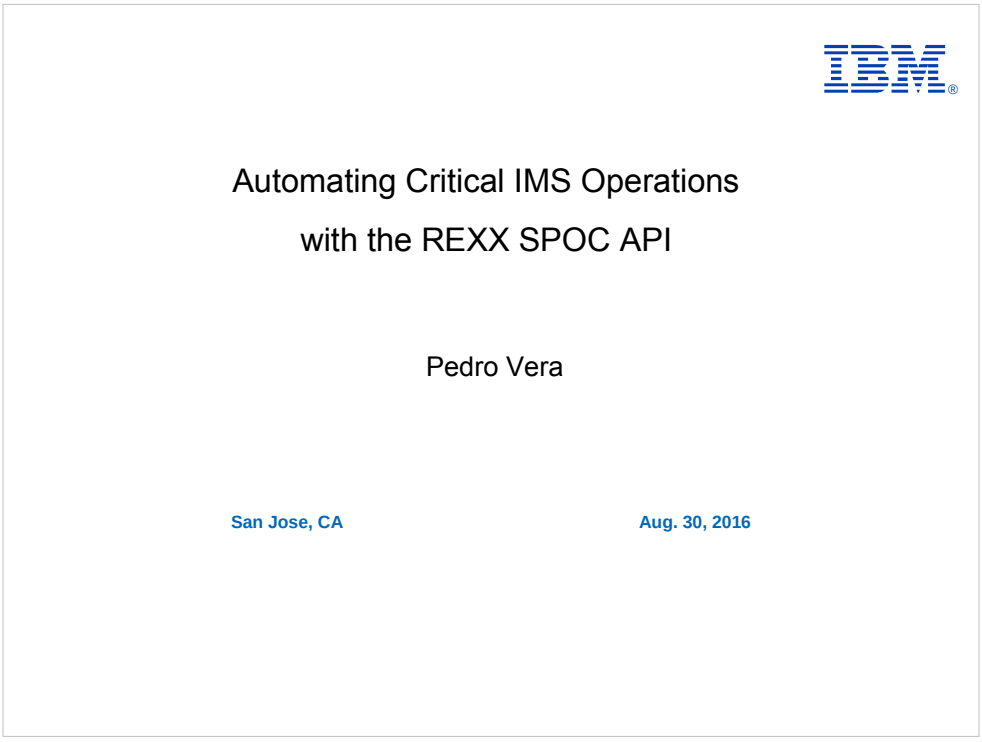


Biography:

I was an MVS system programmer for several years and then in MVS tools support for several years.

Now I work in IMS development, working on ISPF oriented programs.

About Pedro Vera

Frequent contributor to social media
- Ibmmainframes.com - (not part of IBM)
- ibmmainframeforum.com - (not part of IBM)
- TSO-REXX list, ISPF-L list
- Linkedin group moderator: Programming in Rexx

I convinced my IBM colleagues to provide additional rexx support:
- System REXX,
- SDSF REXX API,
- REXX interface for RACF,
- Inline REXX for ISPF panels,
- Inline REXX for ISPF skeletons,
- Trace highlighting

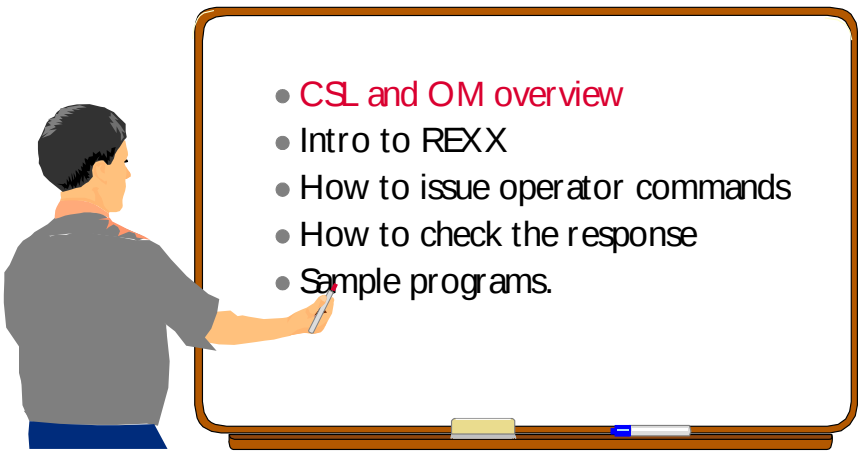
Hello, my name is Pedro Vera
I am a long time IBMer, with about 30 years of working with TSO REXX.

I worked for 15 years in the IMS development group.
After creating the REXX SPOC API for IMS and getting great
customer feedback, I was able to convince my IBM colleagues to provide
Additional rexx support, including:
1. System REXX,
2. the SDSF REXX API,
3. the REXX interface for RACF,
4. inline REXX for ISPF panels,
5. inline REXX for ISPF skeletons.
6. Trace highlighting

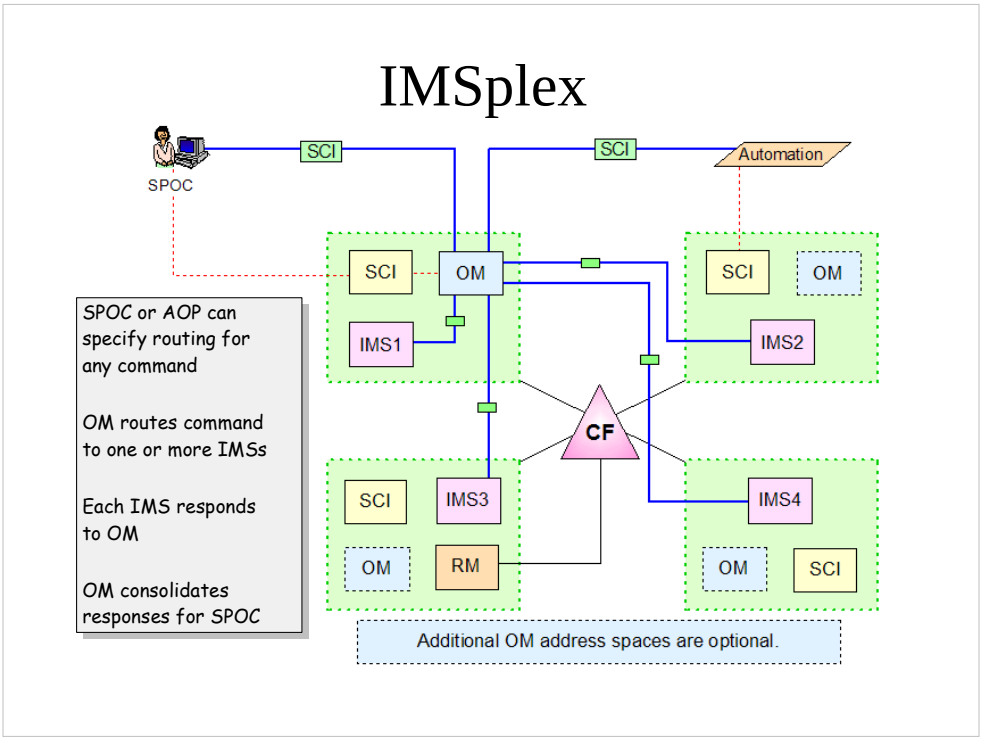
Currently, I am working as a developer for a product called DB2 Administration Tool, which uses rex
heavily.

2

Agenda



I hope to give you enough information so you can go back and try right away.



IMS allows multiple IMS systems to work together as a single image, sharing databases and / or message queues. This is called an IMSplex. The IMS Common Service Layer (CSL) is a collection of address spaces that provide the infrastructure needed for systems management tasks. This includes Operations Manager (OM), Resource Manager (RM), and Structured Call Interface (SCI).

The IMS CSL provides the following:

- Improved systems management
- A single system image
- Ease of use through a single point of control
- Shared resources across all IMS systems

To simplify systems management, the Operations Manager Application Programming Interface (OM API) was added. It allows operator commands to be entered and the responses to be retrieved.

The IMS operator can issue commands to any or all of the IMS subsystems in the IMSplex with a program that utilizes the OM API. That is, the IMSplex can be managed from a single place! This 'Single Point of Control' is referred to as a SPOC.

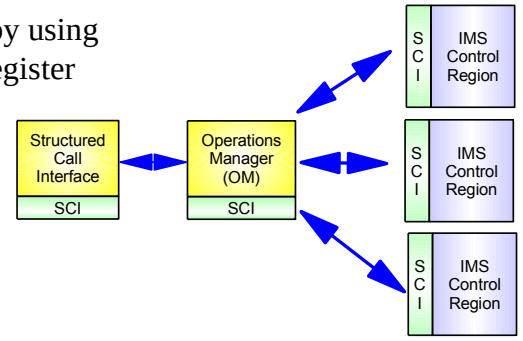
The OM and SCI address spaces are required in order to use Single Point of Control services.

Creating an IMSplex

Started tasks (SCI, OM, RM, and IMS) must have the same name in their startup parameters:

```
IMSPLEX(NAME=PLEX1)   /* Group name=CSLPLEX1 */
```

Others may participate by using CSLSCREG macro to register as part of the IMSplex.



There can be multiple IMSplexes on the system. They are identified by the IMSPLEX parameter in the startup parms. Each component has to specify the same name.

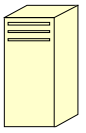
CSLOIxxx
CSLSIxxx
CSLRIxxx

The REXX program has to specify the same IMSplex name.

Operations Manager - API

OM provides an API for

- Automated operations (AO) clients
  - Clients through which commands are entered to OM and then to the command processing client
  - Command may originate from an operator, be received from a network client, or be generated by an automation process
- Command processing (CP) clients
  - Clients which process commands entered from other address spaces
  - IMS is a command processing client
- All OM services are invoked by CSLOMxxx macros
  - Macro coding and use is described in CSL Guide and Reference



API – Application Programming Interface

They provided an assembler API - I provided a REXX API to that invokes a subset of the assembler API.

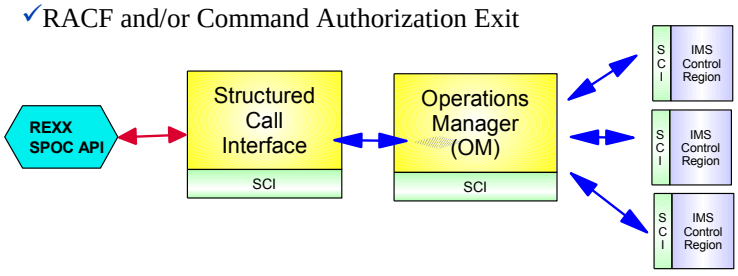
The OM API allows clients to issue IMS operator commands and to get the command responses. New commands (type 2) can only be issued from these new clients. Many of the old commands (type 1) can also be issued from the new clients.

The OM API also allows other vendors to write command processing clients.
Macros are described in a book names Common Service Layer Guide and Reference.

Single Point Of Control (SPOC)

The SPOC is a client of OM

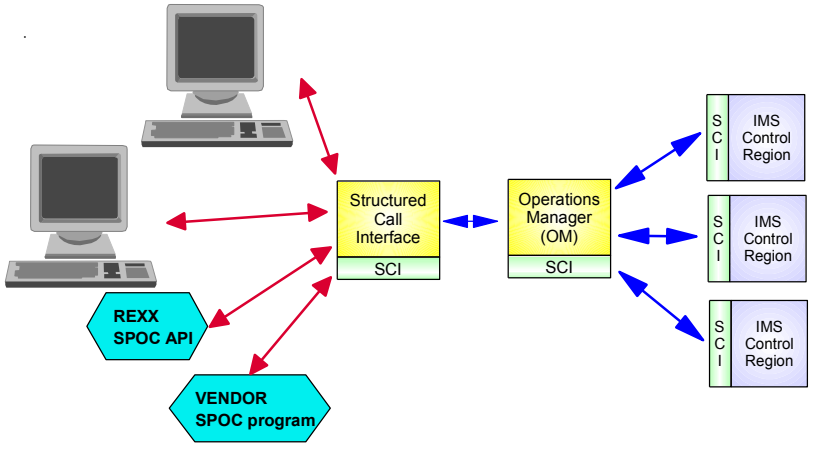
- May or may not be on the same OS as OM
  - ✓Must be on same OS as SCI
  - ✓Uses SCI to communicate with OM
- OM provides security checking
  - ✓TSO userid is used to determine RACF authorization
  - ✓RACF and/or Command Authorization Exit



The Single Point of Control is a client of the Operations Manager.

‘Single’ Does Not Mean ‘Only’

There can be several SPOCs active at any time.



A Single Point of Control (SPOC) means that you can issue commands to all members of an IMSplex at one time. There can be any number of SPOC users active at any time. There is no practical limit to the number of SPOC users.

A SPOC can be a TSO SPOC user, a workstation running the IMS Control Center, a rexx program using the REXX SPOC API, or a vendor product that was written to use the OM API. SCI and OM do not know the difference between the various clients. They receive a command and return a command response. I will talk about the REXX SPOC API in the next session.

The OM API also allows other vendors to write command processing clients.

Commands Supported

- Type-1 commands existed prior to V8. Many can be issued through the OM API.
- Type-2 commands – only supported in OM API environment.

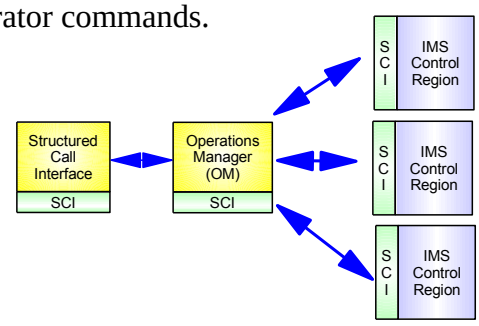
Not all type-1 commands are supported by the OM API. Some abbreviations for commands are not accepted. See Table 15 of Command Reference for list of supported commands.

Type-1 commands existed prior to V8. Type-2 commands will be enhanced to support new function. Type-1 will not normally be enhanced except as needed on a case-by-case basis.

New IMS Commands

IMS now supports new operator commands.

- DELETE
- INITIATE
- QUERY
- TERMINATE
- UPDATE



The new commands are ‘Type-2’ commands and can only be issued through the OM API.

As part of Operations Management enhancements, new commands were added. They support new resources such as LE, OLR, and OLC. Some also support existing resources such as transactions, database, and area.

Support for Type 1 Commands

Previous commands are ‘Type-1’ commands. Most commands are supported through the OM API.

Do not need a command recognition character.

Only certain spellings are accepted through OM API.

DIS ACT	Okay
DIS A	not okay

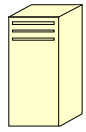


Table 14 in the command reference has a list of supported commands and valid short forms.

Command Security

Commands issued through the OM API are controlled by RACF profiles.

- OPERCMDS class
- Profiles contain the IMSplex name, the command verb, and the resource type.

If your IMSplex name is PLX04 and you want to issue an ACTIVATE NODE command, your userid needs UPDATE authority to:

`IMS.CSLPLX04.ACT.NODE`

Appendix I of the Command Reference has a table of the resource names.

Appendix I of V8 Command Reference is not clear. It has been clarified in the V9 manual. The second node of the resource name is the name of the IMSplex. You should use the same value as is displayed in the QRY IMSPLEX command. The IMSplex name consists of a 'CSL' prefix and a 1-5 character value defined by the customer.
More about command security…

Command Security

- Profiles for display or query commands need READ authority.
- Profiles for commands that change an IMS resource need UPDATE authority.

Reduce the number of RACF profiles by using wildcards:

IMS.CSLPLX04.DIS.*

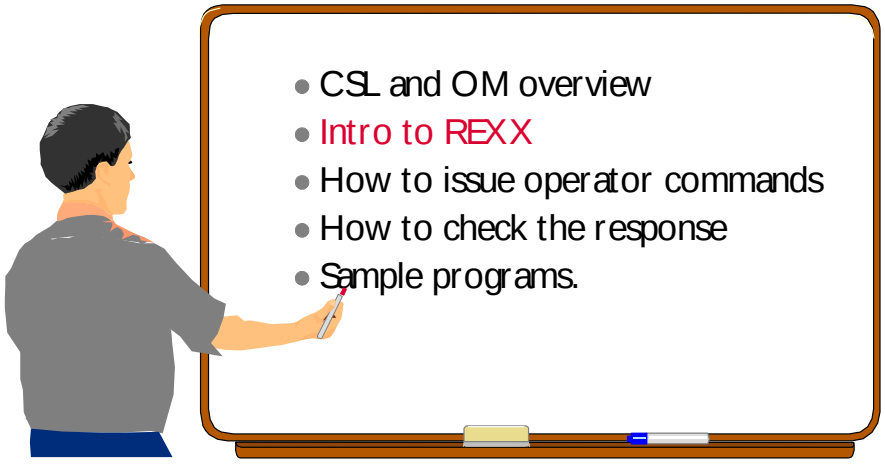
You probably need an all-encompassing profile:

IMS.**

Commands like BROWSE, DISPLAY or QUERY need RACF ‘read’ authority. Other commands, such as those that change a resource need RACF ‘update’ authority to be issued.

Profiles with wildcards will protect several variations of commands.

Agenda



- CSL and OM overview
- Intro to REXX
- How to issue operator commands
- How to check the response
- Sample programs.

Why a REXX SPOC?

Many operator automation programs are written in REXX and run in a NetView environment.

Rexx is the defacto standard for automation.

Rexx is easy to write and debug.

The REstructured eXtended eXecutor language is a versatile, easy to use structured programming language available on numerous platforms.

It is a general purpose programming language like PL/I, or C, and is particularly suitable for:

- Command procedures
- Installation routines
- Application front ends
- User-defined macros (such as editor macros)
- Prototyping
- "Personal computing."

More about the REXX SPOC API.

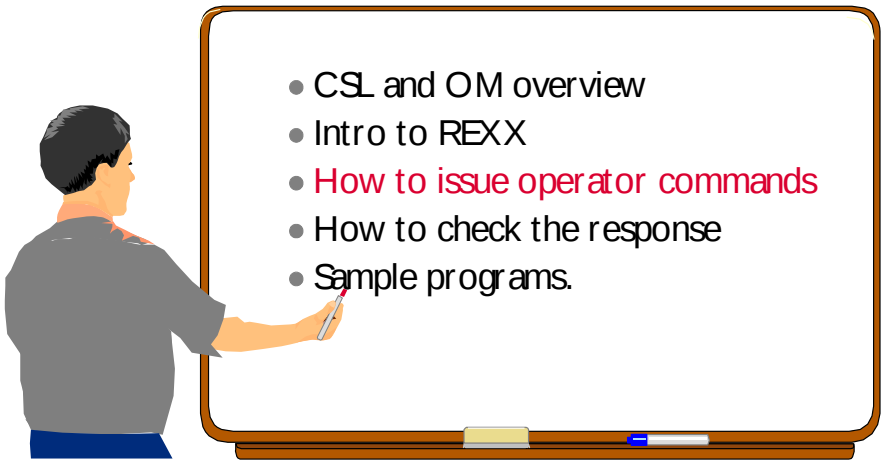
REXX SPOC API

- Runs under TSO or NetView
- May or may not be on the same MVS as OM
- Uses SCI to communicate with OM
- Provides "host command environment" in which IMS commands may be entered to one or more members of an IMSplex
- Saves command responses to a REXX "stem variable" by XML statements.

Since it is REXX, it can do other things as well.

Provides "host command environment" in which IMS commands may be entered to one or more members of an IMSplex

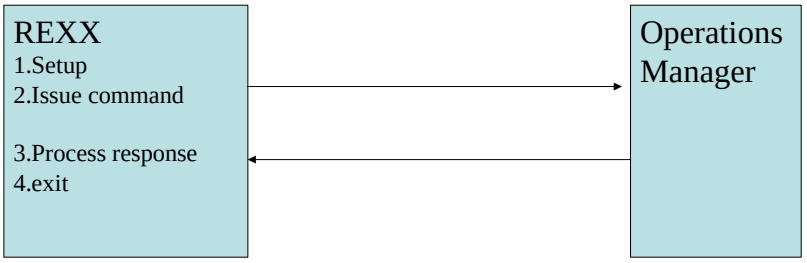
Agenda



- CSL and OM overview
- Intro to REXX
- How to issue operator commands
- How to check the response
- Sample programs.

REXX SPOC API and OM

Functions are provided that utilize SCI and OM services.



You can issue a command

Then process the response when it returns.

REXX SPOC API Sample Program

```
 1  /* rexx */
 2  parse upper arg theIMScmd
 3  Address LINK 'CSLULXSB'
 4  If rc = 0 Then Do
 5     Address IMSSPOC
 6     "IMS plex1"
 7     "ROUTE imsb"
 8     "CART  test12"
 9     "WAIT  3:00"
10     theIMScmd
11     results = cslulgts('resp.','test12','3:15')
12     Say 'imsrc='imsrc   'imsreason='imsreason
13     If resp.0 /= '' Then Do
14         Say resp.0' lines of output'
15         Do indx = 1 To resp.0
16           Say resp.indx
17         End
18       End
19     "END"
20   End
```

This is sample that you can refer to as we go. This is similar to how the normal CONSOLE and GETMSG work. I used them as models.

I will show smaller snippets from this program and explain some of the key points in more detail.

Host Command Environment

Program CSLULXSB sets up the host command environment for rexx. This environment has several subcommands.

Use the 'Address' command to send command processing to particular environments.

Example:

```
Address LINK "CSLULXSB"
Address IMSSPOC
"host command"
```

The rexx host command environment is setup by program CSLULXSB. After it is issued as a program, the IMSSPOC environment is available to the rexx program.

Host commands are typically quoted strings and passed directly to the host command processor.

Setting Preferences

The REXX SPOC API has concepts similar to the ISPF SPOC preferences panel:

```
Address IMSSPOC
 "IMS   plex1"     /* sets IMSplex name     */
 "ROUTE ims3"      /* sets name for explicit
                      route to IMS system   */
 "WAIT  3:00"      /* sets OM timeout value */

   /* program logic here */

 "END"             /* clean up              */
```

Commands IMS, ROUTE, WAIT, CART, and END are supported and perform specific local functions. Anything else is passed to SCI as a command to be performed.

IMS - sets the name of the IMSplex. required.

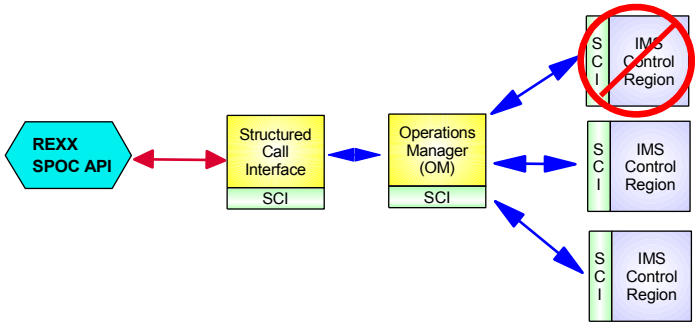
ROUTE - sets the name(s) of the command processors that the will process the command. optional

WAIT - sets the maximum timeout valued for OM to wait for a command response. If the time is reached, OM will return with a 'timed out' return code rather than with complete command response information. If multiple systems are involved, it will return the response from those that responded. optional

END - cleans up control blocks. optional

OM Wait time

1. Command gets routed to all IMSes
2. OM waits for each IMS to respond
3. If any IMS does not respond, OM will wait for a response until the timeout value reached.



WAIT - sets the maximum timeout valued for OM to wait for a command response. If the time is reached, OM will return with a 'timed out' return code rather than with complete command response information. If multiple systems are involved, it will return the response from those that responded.

If one IMS system does not respond when the time expires, OM will send the two responses and also a return code indicating the missing response.

Keeping Track of Commands

The CART host command is used to name the response token.
Function CSLULGTS is used to retrieve the command response

```
Address IMSSPOC
.
"CART    QRYTHETRAN"
"QRY TRAN NAME(V*)"
abc = CSLULGTS("out.","QRYTHETRAN","2:30")
```

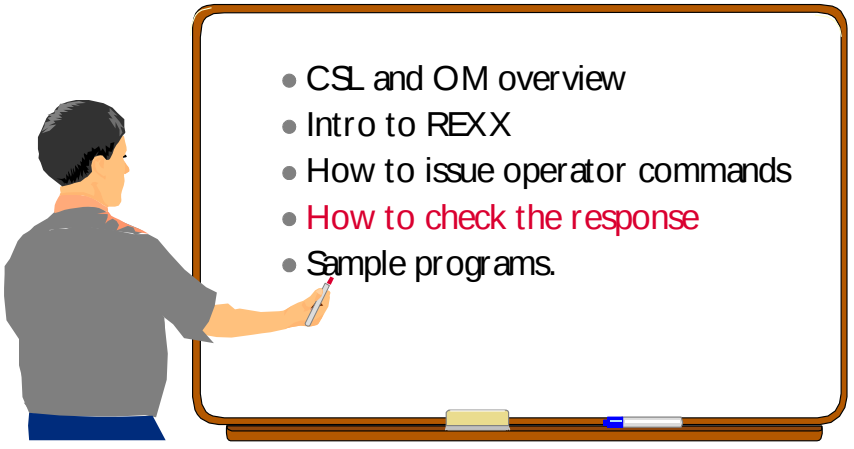
This is another snippet of the program, just further down.

The Command and Response Token is a way to associate a name with the command that will be issued. The token is named using the 'CART' host command. The same token name is used from the CSLULGTS function.

It is an arbitrary name.

The CART is not case sensitive.

Agenda



- CSL and OM overview
- Intro to REXX
- How to issue operator commands
- How to check the response
- Sample programs.

Retrieving the Response

The CSLULGTS function will create a stem variable that will have the command response.

```
abc = CSLULGTS("out.","QRYTHETRAN","2:30")
Do x = 1 To out.0
   Say out.x
End
```

CSLULGTS has three parameters

1. stem name
2. cartid
3. wait time

- stem name is a set of variables in rexx. It is an array. The convention is to set the number of entries in the zero-eth member. out.0 has the number of entries in the array.

- use the same cartid as was previously used in the CART subcommand.

- the wait time is the longest time to wait for the rexx program to wait for the command response. The process is asynchronous, so your program can do something else while the command is executing.

Wait vs. Wait

There are two wait times specified:

- OM Wait time
- REXX wait time

```
Address IMSSPOC
 "WAIT  3:00"      /* OM timeout value */
"QRY TRAN "
abc = CSLULGTS("out.","QRYTHETRAN","2:30")
```

OM wait time – time OM waits for IMS systems to respond. You cannot retrieve missing responses after OM responds.

REXX wait time – it waits for OM response. You can re-try until OM responds.

In the example, you should add logic to retry the CSLULGTS function.

XML Response

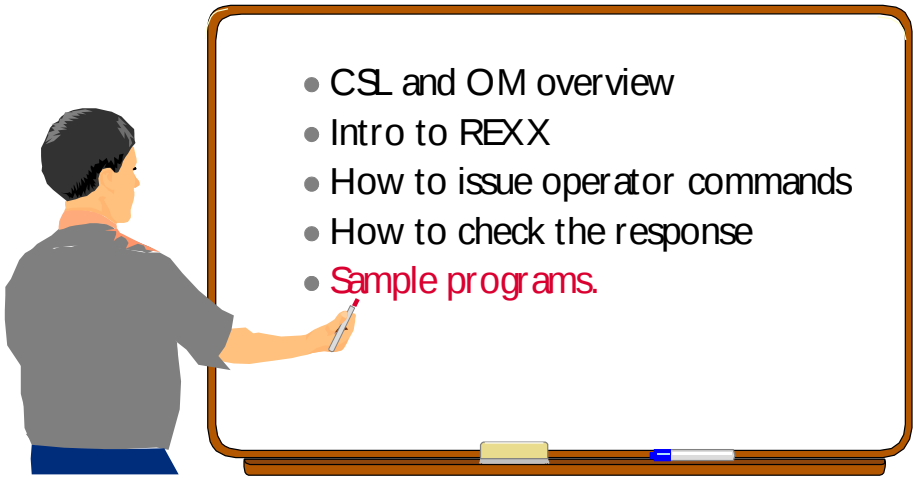
```
abc = CSLULGTS("out.","QRYTHETRAN","2:30")
Do x = 1 To out.0
   Say out.x
End

 out.0 = 89
 out.1 = '<?xml version="1.0"?>'
 out.2 = '<!DOCTYPE imsout SYSTEM "imsout.dtd">'
 out.3 = '<imsout>'
 out.4 = '<ctl>'
 out.5 = '<omname>OM1OM   </omname>'
 out.6 = '<omvsn>1.2.0</omvsn>'
 out.7 = '<xmlvsn>1   </xmlvsn>'
 out.8 = '<statime>2005.105 20:38:52.387088</statime>'
 out.9 = '<stotime>2005.105 20:38:52.475691</stotime>'
 out.10= '<rqsttkn1>QRYTHETRAN      </rqsttkn1>'
 out.11= '<rc>00000000</rc>'
 out.12= '<rsn>00000000</rsn>'
 out.13= '</ctl>'
```

After issuing the CSLULGTS function, your rexx stem variable will have XML statements. The statements will be in the individual elements of the stem variable.

Command responses that are returned through the OM API are embedded in XML tags.

The OM response is intended as a programming interface. See ‘Common Service Layer Guide and Reference’ for more information.

Agenda



- CSL and OM overview
- Intro to REXX
- How to issue operator commands
- How to check the response
- Sample programs.

REXX SPOC API Sample 1

- Program accepts a command as a parameter.
- Issues the command.
- Displays the XML statements.

These are sample programs meant to illustrate capabilities of the REXX SPOC API. You will want to write your programs more robustly.

REXX SPOC API Sample 1

```
 1  /* rexx */
 2  parse upper arg theIMScmd
 3  Address LINK 'CSLULXSB'
 4  If rc = 0 Then Do
 5     Address IMSSPOC
 6     "IMS plex1"
 7     "ROUTE imsb"
 8     "CART  test12"
 9     "WAIT  3:00"
10     theIMScmd
11     results = cslulgts('resp.','test12','3:15')
12     Say 'imsrc='imsrc   'imsreason='imsreason
13     If resp.0 /= '' Then Do
14         Say resp.0' lines of output'
15         Do indx = 1 To resp.0
16           Say resp.indx
17         End
18       End
19     "END"
20   End
```

Highlights of sample program:

- The IMS command picked up in line 2 is executed in line10
- line 5 - sets the default host command process to be IMSSPOC.
- session values are set in lines 6 through 9
- the cartid specified in lines 8 and 11 need to be the same.
- line 10 - the command is a variable. Your implementation could be a fixed valued specified as a quoted string.
- line 11 issues function CSLULGTS to retrieve the command response.
- line 12 displays the return code and reason code
- lines 13 through 18 examine the XML statements returned by CSLULGTS.
- line 19 cleans up IMSSPOC environment.

Return Codes and Reason Codes

Each of the IMSSPOC host commands and the CSLULGTS function set return code values. The values are provided in REXX variables:

imsrc - return code

imsreason - reason code

The values of the variables are character representations of hex values. For example, the imsrc value is c'08000008X' when a parameter is not correct. The character 'x' is at the end of the string so REXX will treat it as a character.

A prefix of '08' is used by automation clients. The return code may also consist of SCI, OM, RM or IMS return codes (not listed here).

Return codes
- "00000000X" Request completed successfully
- "08000004X" Warning
- "08000008X" Parameter error
- "08000010X" Environment error
- "08000014X" System error

Warning reason codes
- "00001000X" command still executing

Parameter error reason codes
- "00002000X" missing or invalid wait value
- "00002008X" missing or invalid IMSplex value
- "00002012X" missing or invalid STEM name
- "00002016X" missing or invalid token name
- "00002020X" too many parameters
- "00002024X" request token not found
- "00002028X" missing or invalid CART value

System error reason codes
- "00004000X" getmain failure

SPOC Sample Job

- JCL to call REXX program.
- Passes command to REXX program.

These are sample programs meant to illustrate capabilities of the REXX SPOC API. You will want to write your programs more robustly.

Sample Job

```
//RXSPOC JOB ,
//   MSGCLASS=H,NOTIFY=USRT002,USER=USRT002
//*
//SPOC     EXEC PGM=IKJEFT01
//STEPLIB  DD DISP=SHR,DSN=IMS.SDFSRESL
//SYSPROC  DD DISP=SHR,DSN=LOCAL.SPOC.REXX
//SYSTSPRT DD SYSOUT=A
//SYSTSIN  DD *
  %REXXSPOC QRY TRAN NAME(V*)
/*EOF
```

The batch job shown is an invocation of the batch TSO command processor. Refer to TSO reference manuals for complete information. Here is a summary of DD name usage:

STEPLIB- the name of the IMS SDFSRESL library

SYSPROC - the name of the rexx library. sysexec can also be used.

SYSTSPRT - output dataset

SYSTSIN - commands to be executed. In this case, it is the name of the member from SYSPROC, plus the parameters.

Sample Program 2

- Query the status of a transaction.
- Examine the command response.
- If the transaction is stopped,
  - Start the transaction.

These are sample programs meant to illustrate capabilities of the REXX SPOC API. You will want to write your programs more robustly.

Sample Program 2

Query the status of a transaction and start it if it is not started.

```
 9   "CART  qrytran12"
10   "QRY TRAN NAME(CDEBTRN3) SHOW(STATUS)"
11   results = cslulgts("resp.","qrytran12","3:15")
12   Do idx = 1 To resp.0
13     parse var resp.idx . "TRAN(CDEBTRN3"    . ,
14                          "MBR(" imsname ")" . ,
15                          "LSTT(" status ")" .
16     If pos('STOSCHD', status) > 0 Then Do
17         "ROUTE" imsname
18         "UPD TRAN NAME(CDEBTRN3) START(SCHD)"
19       End
20   End
```

Highlights of sample program:

- The IMS command is executed in line 10
- line 11 gets the command response
- lines 12 through 15 looks at the XML looking for transaction information.
- line 14 finds the IMS name for which status applies.
- line 16: if status indicates it is stopped, then issue a command to start the transaction.
- line 17 routes the command to the correct IMS (Determined in line 14)
- line 18 starts the transaction.

Sample Program 3

- Query the QCNT of a transaction.
- Examine the command response.
- If QCNT is too high,
  - Start another region.

These are sample programs meant to illustrate capabilities of the REXX SPOC API. You will want to write your programs more robustly.

Sample Program 3

Find tran name in XML and check associated qcnt

```
 9  "CART  qrytran13"
10  "QRY TRAN NAME(SKS1) SHOW(QCNT)"
11  results = cslulgts("resp.", "qrytran13", "3:15")
12  Do idx = 1 To resp.0
13    parse var resp.idx . "TRAN(SKS1" . "Q(" count ")" .
14    If count /> '' &,
15      count  > 5 Then Do
16      "CART  strtrgn05"
17      "START REGION IMSRG5"
18      start? = cslulgts("strt.", "strtrgn05", "10:00")
19      If imsrc = '00000000X' Then
20        Do
21          "CART  updtran14"
22          "UPDATE TRAN NAME(SKS1) SET(CLASS(5))"
24        End
25   End
```

Highlights of sample program:

- the command is issued in line 10
- lines 12 and 13 examine the command response.
- line 13: if information about the transaction is found, determine the QCNT. It is saved to variable 'count'.
- line 15: if the QCNT is too much, resolve the problem
- lines 16 through 18: start a new region to handle more transactions
- lines 20 through 24: change tran class to match the new region.

NOTE: This is just an example of what can be done. In this example, you may end up with less regions running class 5 than are running the previous class.

Sample Program 4

- Query a transaction.
- Get parsed command response.
- Display parts of stem variable

These are sample programs meant to illustrate capabilities of the REXX SPOC API. You will want to write your programs more robustly.

Sample Program 4

```
10  cartid = 'myqry001'
11  "CART" cartid
12  "QRY TRAN NAME(A*)"
13  results = CSLULGTP('qinfo', cartid, '1:30')
14  If qinfo.ctl.rc = 0  Then
15    Do
16      Say "OM name    is" qinfo.ctl.omname
17      Say "Cmd master is" qinfo.cmd.master
18    End
```

In line 13, issue a different function name… it parses the XML, populating

In line 14 -18, examine the stem with fixed tail values.

Examples of stem variable

<omname> </omname>	*stem* .ctl.omname
<rc> </rc>	*stem* .ctl.rc
<input> </input>	*stem* .cmd.input
<rsp> </rsp>	*stem* .rsp.0 (number of rows) *stem* .rsp.x.0 (number of cols) *stem* .rsp.x.y *stem* .rsp.x.y

Examples of stem variable

Messages are in compound array.

<mbr name="membername">	*stem* .msgdata.name.0 (number of systems) *stem* .msgdata.name.y (1 member name)
<msg> </msg>	*stem* .msgdata.msg.y.0 (num of msgs /sys) *stem* .msgdata.msg.y.x (1 message)

41

Examples of stem variable

	stem .report.0 *stem* .report.x	(number of lines) (1 line of report)

Report looks like TSO SPOC display

```
Response for: QRY TRAN SHOW(PSB,QCNT)
Trancode MbrName    CC PSBname       QCnt    LQCnt
ADDINV   IMS2        0                  0
ADDINV   IMS2        0 DFSSAM04                2
ADDINV   SYS3        0 DFSSAM04                1
ADDPART  IMS2        0                  0
```


42

Using REPORT.

- Formatted report
  - Specify the format in the CSLULOPT function

```
ADDRESS LINK 'CSLULXSB'
ADDRESS IMSSPOC; "IMS PLEX1"

“CART QRY1”
“QRY TRAN”
/* get command response */
spoc_rc = CSLULGTP('my.', “QRY1”, "00:10")
"END"

Do x = 1 to my.report.0
  Say my.report.x

End
```

When existing function CSLULGTP is used, it will create a rexx stem variable which contains the formatting ‘listing’

The user specifies the stem prefix.

The stem suffix is always ‘report’ followed by a numeric suffix.

For example:

my.report.0 always has the number of rows in the stem

my.report.1 will have the ‘response for’ line, which contains the command issued.

my.report.2 will have the heading line

my.report.3 will have a dashed line

my.report.4 and on will have the rows of data.

Started Task Example

- Issue Type-2 commands from the system console
- Use started task.
- Started task runs the REXX SPOC API.
- Output is viewable from the MVS syslog.

These are sample programs meant to illustrate capabilities of the REXX SPOC API. You will want to write your programs more robustly.

Sample Started Task

From the operator's console, issue:

```
S IMSCMD,CMD='QRY TRAN NAME(ABC)'
```

Started task JCL:

```
//IMSCMD   PROC CMD='QRY IMSPLEX'
//SPOC     EXEC PGM=IKJEFT01,
//  PARM='%RXCMD &CMD'
//STEPLIB  DD DISP=SHR,DSN=IMS.SDFSRESL
//SYSPROC  DD DISP=SHR,DSN=SVL.SPOC.REXX
//SYSTSPRT DD SYSOUT=A
//SYSTSIN  DD DUMMY
```

The JCL shown is an invocation of the batch TSO command processor. Refer to TSO reference manuals for complete information.

The PROC statement defines a default command. The default is overridden by the operator who types the command at the system console.

Here is a summary of DD name usage:

STEPLIB- the name of the IMS SDFSRESL library

SYSPROC - the name of the rexx library. sysexec can also be used.

SYSTSPRT - output dataset

SYSTSIN - commands to be executed. In this case, it is not used! The command to execute is the EXEC statement parameter.

Started Task Program

```
 1  /* rexx */
 2  parse upper arg theIMScmd
 3  Address LINK 'CSLULXSB'
 4  If rc = 0 Then Do
 5     Address IMSSPOC
 6     "IMS plex1"
 7     "CART  test12"
 8     "WAIT  3:00"
 9     theIMScmd
10     results = cslulgts('resp.','test12','3:15')
11     If resp.0 /= '' Then Do
12       Do indx = 1 To resp.0
13         Address TSO,
14            "SEND '"resp.indx"' CN(0)"
15         End
16       End
17     "END"
18   End
```

Highlights of sample program:

- The IMS command picked up in line 2 is executed in line 10
- line 9 - the command is a variable. Your implementation could be a fixed valued specified as a quoted string.
- line 10 issues function CSLULGTS to retrieve the command response.
- lines 12 through 18 examine the XML statements returned by CSLULGTS.
- lines 13 and 14 use the SEND command to send the response to the operators console and to SYSLOG.

Setting up Host Command

- Call IRXSUBCM service routine to set up host environment
- Program CSLULRXX gets control

```
Declare
  function                char(8),
  1 environ_str           like(SUBCOMTB_ENTRY),
  environ_ptr             pointer(31),
  environ_len             fixed(31),
  cmd_environ             char(8),
  subcm_code              fixed(31);
```

```
%PARMBLOCK_SUBCOMTB = 'subcom_ptr';
%INCLUDE SYSLIB(irxsubct);
```

```
      function                             = 'ADD';
      environ_str.SUBCOMTB_NAME            = "IMSSPOC ";
      environ_str.SUBCOMTB_ROUTINE         = "CSLULRXX";
      environ_str.SUBCOMTB_TOKEN(1:4)      = char(gblptr);
      environ_ptr                          = addr(environ_str);
      environ_len                          = length(environ_str);
      cmd_environ                          = "IMSSPOC ";
- - - - - - - - - - - - - - - - - - - -  9 Line(s)
      Call IRXSUBCM( function,
                     environ_ptr,
                     environ_len,
                     cmd_environ,
                     envbptr
                     );
```

Call the IRXSUBCM service routine to set up a host command environment

In this example, program CSLULRXX gets control.

Processing Host Command

- Program CSLULRXX gets control
- Examine cmd_ptr -> string area

```
000136  CSLULRXX: proc(hostcmd, cmd_ptr, cmd_len,
000137                 token_ptr, cmd_code)
000227  Declare
000228    hostcmd        char(8),
000229    cmd_ptr        pointer(31),
000230    cmd_len        fixed(31),
000231    token_ptr      pointer(31),
000232    cmd_code       fixed(31),
000501    cmd_code  = rexx_retcode;
000502    return code(rexx_retcode,rexx_rsncode);
000503
```

In this example, program CSLULRXX gets control.

The parameter list includes:

Host command

A Pointer to the Command string

The length of the command string

A token to save your workareas

And the return code from the host command.