z/OS System REXX

December 2010

Ron Northrup
Email Address: rnorthru@us.ibm.com

Updated 11/30/2010 – Harris Morgenstern
Email Address: hmorgen@us.ibm.com

Table of Contents

- Trademarks
- Theme
- Session Objectives
- Overview
- Logical Overview Diagram
- Comparison of REXX & TSO Servers
- TSO/E Command Support
- Security & Enclave Considerations
- Usage & Invocation
- Arguments & Variables
- Interaction & Dependencies
- Migration & Coexistence Considerations
- Installation
- Exploiting SYSREXX
- Best Practices
- Session Summary
- Appendix

Theme

Simplification

The role of SYSREXX in “New Face of z/OS” is to provide an infrastructure through which REXX execs may be run outside the normal TSO/E or Batch environments, using a simple programming interface. This enables the leveraging of base operating system components by new style applications that will, over time, lead to simplified interaction and more intuitive system management capabilities on z/OS.

The ability to initiate REXX execs directly from an operator console has been long overdue on z/OS and is a drag along benefit of this initiative.

The possibilities for exploiting existing REXX code through the use of SYSREXX are vast, whether to provide operator assists or to provide routines that can be leveraged by new strategic initiatives.

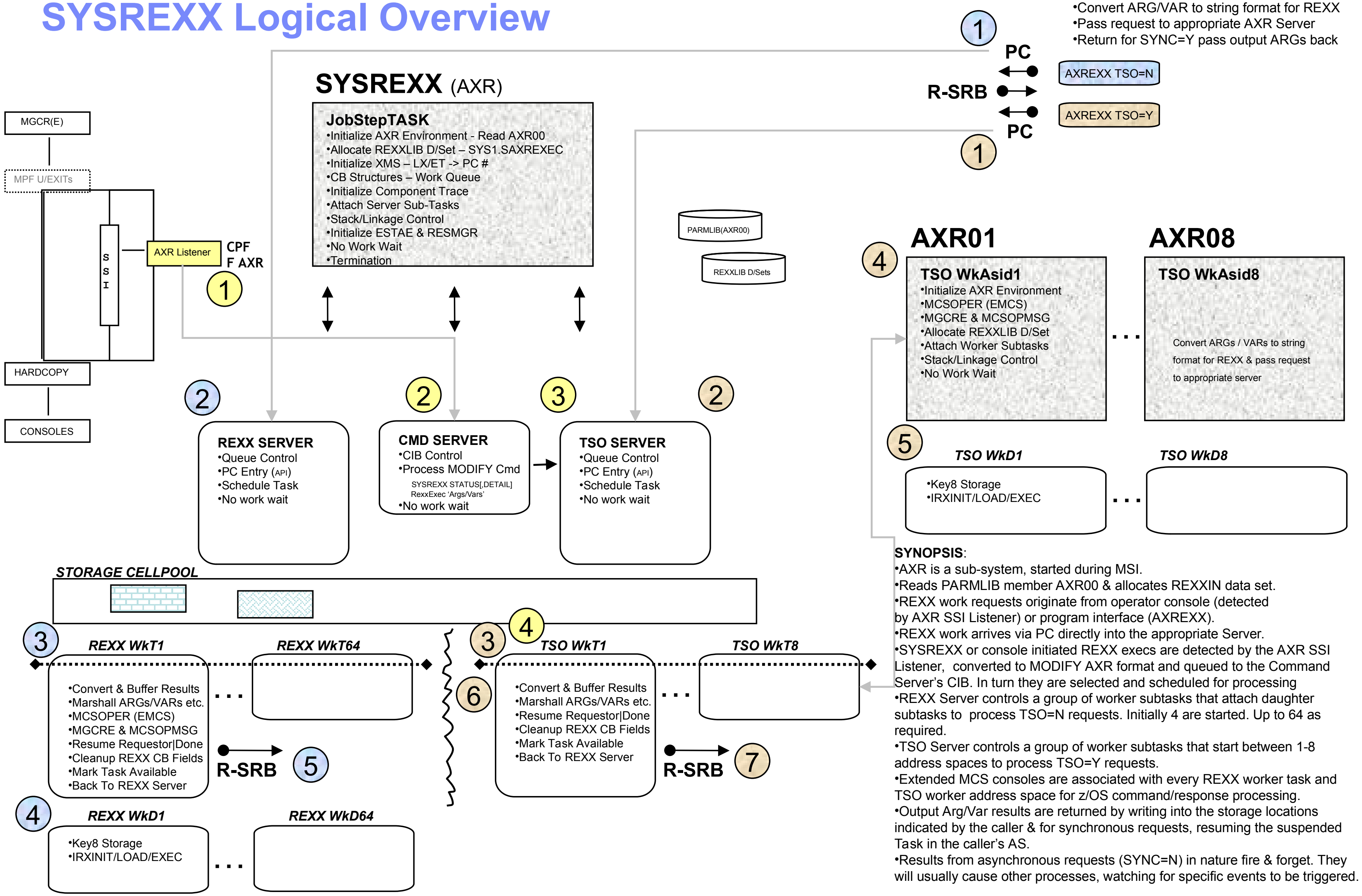
Session Objectives

- **Describe System REXX**
  - ***The Design***
  - ***Customization***
  - ***Getting Started***
  - ***Where to find more information***

Overview

- **Problem Statement / Need Addressed:**
  - *Required an Infrastructure to Support Web Based Initiatives interacting with z/OS Components*
    - New Face of z/OS
      - z/OS Simplification & Management
- **Solution:**
  - *SYSREXX allows execs to be run simply & independently from traditional TSO/E & Batch*
    - Invocation uses a single program interface (AXREXX)
    - Operator exploitation directly from a console
- **Benefit:**
  - *Enables rapid development & deployment of system programmer tools and operator assists*
  - *Can be exploited by new & old style applications*
  - *Health Checks may be written in REXX*

SYSREXX Logical Overview



- Convert ARG/VAR to string format for REXX
- Pass request to appropriate AXR Server
- Return for SYNC=Y pass output ARGs back

SYSREXX (AXR)

JobStepTASK
- Initialize AXR Environment - Read AXR00
- Allocate REXXLIB D/Set – SYS1.SAXREXEC
- Initialize XMS – LX/ET -> PC #
- CB Structures – Work Queue
- Initialize Component Trace
- Attach Server Sub-Tasks
- Stack/Linkage Control
- Initialize ESTAE & RESMGR
- No Work Wait
- Termination

REXX SERVER
- Queue Control
- PC Entry (API)
- Schedule Task
- No work wait

CMD SERVER
- CIB Control
- Process MODIFY Cmd

SYSREXX STATUS[,DETAIL]
RexxExec 'Args/Vars'
- No work wait

TSO SERVER
- Queue Control
- PC Entry (API)
- Schedule Task
- No work wait

AXR01

TSO WkAsid1
- Initialize AXR Environment
- MCSOPER (EMCS)
- MGCRE & MCSOPMSG
- Allocate REXXLIB D/Set
- Attach Worker Subtasks
- Stack/Linkage Control
- No Work Wait

AXR08

TSO WkAsid8

Convert ARGs / VARs to string format for REXX & pass request to appropriate server

TSO WkD1
- Key8 Storage
- IRXINIT/LOAD/EXEC

TSO WkD8

STORAGE CELLPOOL

REXX WkT1
- Convert & Buffer Results
- Marshall ARGs/VARs etc.
- MCSOPER (EMCS)
- MGCRE & MCSOPMSG
- Resume Requestor|Done
- Cleanup REXX CB Fields
- Mark Task Available
- Back To REXX Server

REXX WkT64

REXX WkD1
- Key8 Storage
- IRXINIT/LOAD/EXEC

REXX WkD64

TSO WkT1
- Convert & Buffer Results
- Marshall ARGs/VARs etc.
- Resume Requestor|Done
- Cleanup REXX CB Fields
- Mark Task Available
- Back To REXX Server

TSO WkT8

SYNOPSIS:
- AXR is a sub-system, started during MSI.
- Reads PARMLIB member AXR00 & allocates REXXIN data set.
- REXX work requests originate from operator console (detected by AXR SSI Listener) or program interface (AXREXX).
- REXX work arrives via PC directly into the appropriate Server.
- SYSREXX or console initiated REXX execs are detected by the AXR SSI Listener, converted to MODIFY AXR format and queued to the Command Server's CIB. In turn they are selected and scheduled for processing
- REXX Server controls a group of worker subtasks that attach daughter subtasks to process TSO=N requests. Initially 4 are started. Up to 64 as required.
- TSO Server controls a group of worker subtasks that start between 1-8 address spaces to process TSO=Y requests.
- Extended MCS consoles are associated with every REXX worker task and TSO worker address space for z/OS command/response processing.
- Output Arg/Var results are returned by writing into the storage locations indicated by the caller & for synchronous requests, resuming the suspended Task in the caller's AS.
- Results from asynchronous requests (SYNC=N) in nature fire & forget. They will usually cause other processes, watching for specific events to be triggered.

REXX Server vs TSO Server

REXX Server

- Execs run in AXR AS
  - *1-64 Worker Subtasks*
  - *EMCS for each subtask*
  - *Detach after 100 execs*
  - *Default HCE - MVS*
  - *Recommend no Data Set Allocation here*
- Availability impact if something serious breaks

TSO Server

- Execs run in AXRnn AS
  - *1-8 TSO Server Address Spaces*
  - *EMCS for each AS*
  - *Detach after 200 execs*
  - *All Allocation capabilities*
  - *Default HCE - TSO*
    - IKJTSOEV – Dynamic TSO environment
    - SYSCALL supported
- Better availability should something serious break.. just cancel AXRnn AS

TSO/E Command Support

1 of 3

- **TSO=YES requires an isolated address space to avail itself of TSO/E services and POSIX host commands**
  - ***authorized … jscbauth = 1***
  - ***Address spaces called AXR01-08***
  - ***within a dynamic TSO/E environment … ikjtsoev service***
    - **TSO/E was modified to allow Sysrexx execs to invoke authorized commands**

TSO/E Command Support

2 of 3

- **TSO=Y Requests Run**
  - ***authorized … jscbauth = 1***
  - ***in a separate address space … axr01-08***
  - ***within a dynamic TSO/E environment … ikjtsoev service***
- **We restrict the number of commands that Sysrexx officially supports because the Dynamic TSO environment does not set up all the features of the Terminal Monitor Program (TMP)**
- **Supported Commands**
  - ***allocate attrib***
  - ***call delete exec***
  - ***free help profile***
  - ***rmm smcopy time***

TSO/E Command Support

3 of 3

- **Problems Encountered:**
  - ***Some authorized commands do not tolerate a dynamic environment***
    - **TSO/E control blocks are unavailable or incomplete**
      - CONSOLE host command environment is not supported
    - **Userid function does not return a valid userid**
      - In R11 (or with OA26802 applied), in most cases this has been corrected, but still would be in the case where the AXREXX invoker does not specify (or default to) a valid security environment.
  - ***Link, Load, or Attach unauthorized modules … ABEND306***
  - ***Resources not cleaned up, may influence subsequent execs***
    - **Data sets left allocated are freed (but execs should clean up)**

Security & Enclave Considerations

- **AXREXX is an <u>Authorized</u> System Service**
  - *Security Controls Essential*
    - **Userid and Groupid Associations for SYSREXX Address Spaces**
    - **Access to APF Library**
    - **Permissions**
      - Standard security administration
        - > what can be accessed and/or run

- **EXECs by default use Invoker’s Security Environment**
  - *Alternatively may use:*
    - **Authority of a 3rd party – a surrogat**
    - **Special userid which is assigned to AXRUSER**

- **EXECs use Invoker’s Enclave service class**
    - **Prevent CPU priority inversion & excessive resource usage**
    - **Resource usage is charged back to invoker**

System REXX Address Spaces

- **Some Installation’s Tightly Control Started Procedures**
- **Security Administrator must update the STARTED class profile and/or the Started Procedures Table (ICHRIN03)**
- **This assigns Userid and Group characteristics to these address spaces**
- **The AXR address space is System REXX**
  - ***Attach to Group for System address spaces***
  - ***Requires READ authority***
    - **SYS1.PARMLIB & SYS1.SAXREXEC**
    - **In R11 (or with OA26802 applied) - REXXLIB data sets**

Usage & Invocation

1 of 7

- Easy way for Web Based Servers to run commands/functions & get back pertinent details
- Method for system and application components to exploit REXX parsing strengths
- Leverage REXX coding skills which are extensive & span all operating systems
- Faster development possibilities for new Health Checks
- Controlling RACF passwords and phrases

Usage & Invocation

2 of 7

- Provision of simplified operator assist functions & quick fixes when necessary
- Exploitive possibilities by Customer code, IBM & ISV components & products

- SYSREXX is a Server
  - *Starts early during Master Scheduler Initialization*
  - *No STOP capability but the AXR address space can be forced*
    - FORCE AXR,ARM

Usage & Invocation

3 of 7

- **New Macro is AXREXX**
  - *Caller must be authorized*
  - *Two Modes of Execution*
    - TSO=NO Limited Data Set Support
    - TSO=YES Full Data Set Support Permitted
  - *Three Operations are supported*
    - REQUEST=EXECUTE | CANCEL| | GETREXXLIB
  - *Arguments & Variables are used to pass input to and receive output from the REXX exec*
  - *Parameter List is mapped by the AXRZARG macro*

Usage & Invocation

4 of 7

- **New Macro is AXREXX** continued
  - *Data Types supported:*
    - Input is converted to strings
    - Output is returned as signed/unsigned, char, binary, hex
  - *Security Environment of Requester is used by default*
    - Can be changed with SECURITY= keyword
  - *Service Classification using Requester's enclave*
- **Time Limitations are Applied to Execs**
  - *Default is 30 seconds*
- **Cancellation of REXX Exec using API**
  - *OREQTOKEN parameter with REQUEST=CANCEL*

Usage & Invocation

5 of 6

- Operator Command Support
  - *Detected by System REXX SSI Listener*
    - Uses a customizable Command Prefix (cpf)
  - *Administrative Support*
    - cpfSYSREXX STATUS[,DETAIL]
    - cpfSYSREXX REXXLIB
  - *Seamless operator interface for REXX execs*
    - cpfRexxExecName Arg1 Arg2 … Argn
- REXX Execs Read From SYS1.SAXREXEC
  - *Or other data set in REXXLIB Concatenation (R11 or OA28602 applied)*

Usage & Invocation

6 of 7

- **Originally three SYSREXX Functions were provided:**
  - ***AXRCMD***

    Issue a console command and return command responses
  - ***AXRWTO***

    Issue a single line message to a console
  - ***AXRMLWTO***

    Issue a multiline line message to a console

Console Interface for REXX Execs

1 of 4

- **The Console Interface is the MODIFY AXR command**
  - *Can be replaced by the CPF value defined in AXR00*
- **Provides a seamless operator interface for REXX execs**

MODIFY AXR,<execname>[,TIMEINT=<seconds> <argument>]

cpf<execname[,TIMEINT=<seconds> <argument>]

- **REXX execs are run with the AXREXX macro CONSDATA keyword in effect**
  - *AXRWTO and AXRMLWTO functions send messages to the invoking console as command responses*

Console Interface for REXX Execs

2 of 4

- Command Authority determines which commands may be entered from a console.
- A userid logging onto a console has all permissions associated with the userid
- Console initiated REXX Execs run TSO=YES
- A blank separates arguments from one another and from the other command keywords
- The <argument> is a CHAR string consisting of the remainder of the command line
- SAY, TRACE and REXX messages are directed to the invoking console as multi-line message AXR0500I

Console Interface for REXX Execs

3 of 4

- **AXRUSER has NOTHING to do with the Console Interface for REXX Execs**
  - ***Common misconception***
  - ***Security environment is that of the user logged onto the console***
- **MVS.SYSREXX.EXECUTE.**
  - ***Resource used to determine whether the invoker is authorized to execute the exec***
- **MVS.SYSREXX.**
  - ***Resource used to determine whether the invoker is authorized to issue F AXR,SYSREXX STATUS and other MODIFY AXR commands***

Console Interface for REXX Execs

4 of 4

- **MODIFY AXR,SYREXX STATUS**
  - *Provides overall status of System REXX*

- **MODIFY AXR,SYSREXX STATUS,DETAIL**
  - *Provides detailed information about actively running execs*

- **MODIFY AXR,SYSREXX REXXLIB (R11 or OA28602 installed)**
  - *Displays the REXXLIB concatenation*

Arguments & Variables

- **Up to 20 ARGs and 256 VARs are Supported**
- **Caller specifies the data type of the ARG/VAR, the name of the ARG/VAR and whether it is input, output or both**
  - ***Signed/Unsigned 32 or 64 bit, CHAR String, BIT String, and HEX String Types are supported***
- **If the ARG/VAR is input, System REXX will convert the data type to a character string and set the ARG/VAR prior to exec invocation**
- **If the ARG/VAR is output, System REXX will obtain the final value of the ARG/VAR, convert it into the specified data type and copy it into the AXREXX invokers storage**
- **AXRZARG macro provides the ARG/VAR list mapping**

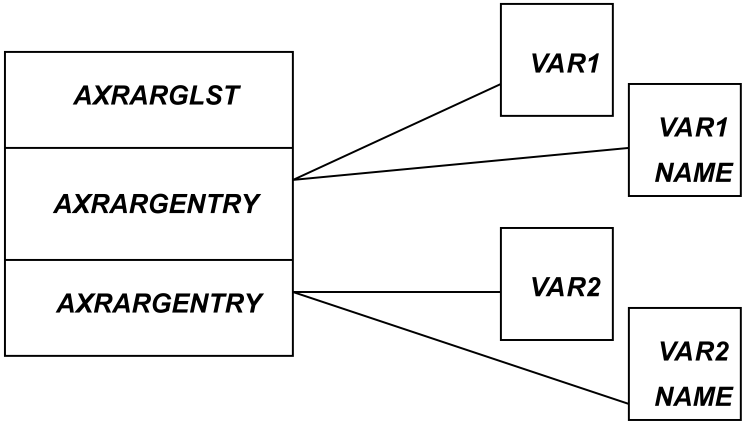
- **Special Variables set by AXR on input to the exec**

• **AXRREQTOKEN**	**32 Hex**	**Request Token**
• **AXRINDD**	**8 Char**	**If RexxInDsn Specified**
• **AXROUTDD**	**8 Char**	**If RexxOutDsn Specified**

AXRZARG Mapping Macro

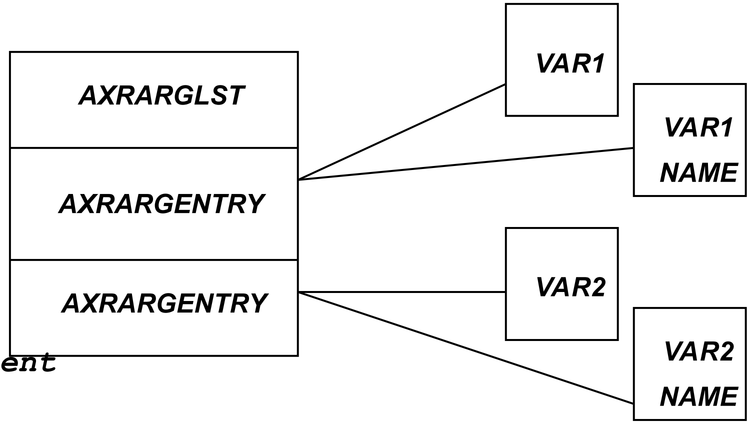
- **AXRARGLST**

```
AXRARGLST                DSECT
AXRARGLSTID              DS   CL4    Use AxrArgLstAcro for REXXArgLst and    X
                                     AxrVarLstAcro for REXXVarLst
AXRARGLSTVER             DS   X
AXRARGLSTRSV1            DS   CL3    Reserved - must be 0
AXRARGLSTNUMBER          DS   H      Number of arguments
AXRARGLSTENTRYINERROR    DS   H      Output argument in error
AXRARGLSTRSV2            DS   CL4    Reserved - must be 0
AXRARGLSTEND             DS   0C
AXRARGLSTACRO            EQU  C'ARGL'
AXRVARLSTACRO            EQU  C'VARL'
AXRARGLSTCURVER          EQU  0
AXRARGLSTVER0            EQU  0      Version 0
AXRARGLST_LEN            EQU  *-AXRARGLST
```



AXRZARG Mapping Macro

- **AXRARGENTRY**



```
AXRARGENTRY          DSECT
AXRARGADDR           DS   AD     Address of argument
                     ORG  AXRARGADDR
AXRARGADDRHIGH       DS   F      High half of address
AXRARGADDRLOW        DS   A      Low half of address
AXRARGNAMEADDR       DS   AD     Address of the name of the argument.   X
                                 this is required for all variables and X
                                 output arguments
                     ORG  AXRARGNAMEADDR
AXRARGNAMEADDRHIGH   DS   F      High half of address
AXRARGNAMEADDRLOW    DS   A      Low half of address
AXRARGLENGTH         DS   F      Length of arg/var in bytes
AXRARGALET           DS   F      Alet of arg/var
AXRARGNAMEALET       DS   F      Alet of name of arg/var
```

Interactions & Dependencies

- Hardware Dependencies
  - *None*
- Software Dependencies
  - *To Process Compiled REXX Code*
    - IBM Library for REXX on zSeries OR
    - IBM Library for REXX on zSeries Alternate Library
- Exploiters
  - *CIM Server .. Cluster Instrumentation*
  - *Health Checker*
  - *Security Server (RACF)*
  - *Open to exploitation by IBM, ISV and Customer Code*

Installation

1 of 5

- No Installation Requirements on z/OS V1 R9
  - *Install OA26802 (no need if running z/OS V1 R11 or higher)*
  - *SYSREXX is a new BCP Component .. 5752SCAXR*
  - *All Cross-Component Dependencies Included In Base*
- Implementation & Customization
  - *Easy & Straightforward*
  - *Reference Publications*
    - Authorized Assembler Services Guide .. SA22-7605
- Starting SYSREXX Requires
  - *Ability to read SYS1.PARMLIB*
  - *Parmlib member CTIAXR00*
  - *Ability to read SYS1.SAXREXEC (and the REXXLIB)*

Installation

2 of 5

- **Parmlib Support**
  - ***SYSREXX Customization: AXR00***
    - **In R11 (or with OA28602 installed) IEASYSxx can specify an AXRxx parmlib concatenation**
      - AXR=(xx,yy,…)
      - First valid instance of option used
    - **CPF('cpfvalue',SYSTEM|SYSPLEX)**
    - **AXRUSER(userid)**
      - Optional with APAR OA22146 applied

Installation

3 of 5

- Parmlib Support (continued)
  - *REXXLIB support (R11 or OA28602 installed)*
    - Up to 255 data sets are supported
    - SYS1.SAXREXEC concatenated as the last data set unless specified
    - Data sets must have the same characteristics as SYS1.SAXREXEC (i.e. variable length records of size 255)
  - *Component Trace (SYSAXR): CTIAXRxx*
    - Defaults to ERROR tracing

Installation

4 of 5

- **Proclib Support**
  - *SYSREXX is restarted using procedure AXRPSTRT*
    - **S AXRPSTRT**
      - Uses AXR00 for start parms
    - **S AXRPSTRT,AXR=(xx,yy…) - in R11 or OA28602 installed**
- **REXX LIBRARY/REXX ALTERNATE LIBRARY**
  - *Must be set up to run any compiled exec*

Installation

5 of 5

- **Parmlib & Proclib Members**
  - ***Copy member AXR00 from SAMPLIB to PARMLIB***
  - ***Customize AXR00***
    - **Choose a Command Prefix**
      - `CPF('@',SYSTEM)`
    - **If required setup a surrogat userid that SYSREXX may use**
      - `AXRUSER(SYS1)`
    - **On R11 or with OA28602 installed set up REXXLIB**
  - ***On R11 or with OA28602 installed set up IEASYSxx and specify multiple AXRxx members***
  - ***CTRACE member is CTIAXR00***
  - ***Restart AXR using the AXRPSTRT procedure***

Exploiting SYSREXX

- **Customer Validation Is Critical**
  - ***Implement automation tasks that could be carried out from SYSREXX:***
    - **Perhaps launch from exits**
  - ***Implement operator assists that could be launched from the console via SYSREXX***
  - ***Implement health checks in SYSREXX***
    - **Reference Session 2815 from Orlando SHARE Proceedings**

      ”z/OS 1.9: Health Checker Update and User Experiences with System REXX Checks”

      http://ew.share.org/proceedingmod/abstract.cfm?abstract_id=17633

Exploiting SYSREXX

- **Invocation from the AXREXX macro**
  - ***SAY/TRACE output goes to the output data set specified by REXXOUTDSN on the AXREXX invocation***
    - **If not specified output goes to the console specified by CONSDATA (this is the invoking console when the exec is initiated from an operator console)**
      - If CONSDATA is not specified, SAY/TRACE output is dropped
  - ***AXRWTO/AXRMLWTO output goes to the Console specified by CONSDATA. Goes to the System Log if CONSDATA is not specified***

Exploiting SYSREXX

- Invocation from the AXREXX macro
  - *Errors converted into return/reason codes*
    - More info is needed to debug problems
    - REXX communicates with messages NOT codes
  - *Use of REXXDIAG keyword essential for SYNC=YES*
    - System REXX processing interrogates REXX messages and returns REXX and TSO message ids
    - In the case of a Syntax/Run-time error, returns the line where the error occurred
    - Contents of REXXDIAG are reason code specific

Exploiting SYSREXX

- **Invocation from the AXREXX macro**
  - ***Diagnostics when failing to process an input/output argument or variable***
    - **For example, in the REXXVARS AXREXX parm, MyVar is an output variable of type AXRARGTypeUnSigned, but the variable contains non-numeric data**
    - **Bad argument/variable number is stored in AXRARGLstEntryInError**
  - ***SYNC=YES option will suspend invoker until exec completes***
    - **Do not use if the invoker cannot tolerate a delay (e.g. in an MPF exit)**

Exploiting SYSREXX

- **Sample invocation of AXREXX macro**
- **Takes a jobname as input and issues the DISPLAY JOBS command to obtain the ASTE address of the job**
  - ***Typical of some applications that have been written in support of z/OSMF***
    - **Invoke a command to fetch some piece of data to return to the invoker**
  - ***Many execs also invoke utility programs to obtain data in other ways e.g. generating data sets***
    - **For example EREP can easily be invoked from an exec to format SYS1.LOGREG**

```
IEE115I 14.45.42 2010.347 ACTIVITY   FRAME  1     F      E   SYS=SY1
 JOBS     M/S    TS USERS    SYSAS    INITS   ACTIVE/MAX VTAM     OAS
00000    00005    00000      00026    00009    00000/00300       00003
 CONSOLE  CONSOLE           NSW  *   A=0009   PER=NO   SMC=000
                                     PGN=N/A  DMN=N/A  AFF=NONE
                                     CT=000.152S  ET=132.523S
                                     WKL=SYSTEM   SCL=SYSTEM   P=1
                                     RGP=N/A      SRVR=NO  QSC=NO
                                     ADDR SPACE ASTE=0F255240
                                     DSPNAME=IEAM0015 ASTE=0C032780
```

Sample output for the Display JOBS,<jobname> command

```
GETASTE  CSECT ,
GETASTE  AMODE 31
GETASTE  RMODE 31
*
*  TITLE: GETASTE
*
*  FUNCTION:  OBTAIN THE ADDRESS OF MASTER'S ASTE BY INVOKING
*     AN EXEC TO PARSE THE OUTPUT OF DISPLAY JOBS,*MASTER*.
*     THE FOLLOWING EXEC TAKES A JOBNAME AS AN INPUT ARGUMENT
*     AND SETS THE VARIABLE OUTASTE@.
*  INPUT:   InJobname (argument)
*  OUTPUT: OutAste@ (Output variable)
*
* NUMERIC DIGITS 25
* ARG INJOBNAME
* MYCMD = 'D JOBS,' || STRIP(INJOBNAME);
* RESULT =  AXRCMD(MYCMD,OUTPUTVAR.,10); /* Issue the
*          cmd to obtain info about the address space*/
* IF RESULT = 0 THEN
*   DO;
*     OUTASTE@ = ' '
*     DO LINENUM = 1 TO OUTPUTVAR.0 WHILE(OUTASTE@=' ');
*       PARSE VAR OUTPUTVAR.LINENUM 'ASTE=' OUTASTE@
*     END;
*     IF OUTASTE@ = ' ' THEN
*       DO;
*         MYRETCODE = 8;
*         OUTASTE@ = 0;
*       END;

*     ELSE
*          MYRETCODE = 0;
*   END;
* ELSE
*   DO;
*     MYRETCODE = 12;
*     OUTASTE@ = 0;
*   END;
* EXIT MYRETCODE;
*******************************************************************
         BAKR  14,0
         USING GETASTE,12
         LR    12,15
```

```
 MODID BR=YES
 XC   MYARGLST,MYARGLST    CLEAR THE ARGLST HEADER
 XC   MYVARLST,MYVARLST    CLEAR THE VARLST HEADER
 XC   MYARGEN1,MYARGEN1    CLEAR THE ARG ENTRY
 XC   MYVAREN1,MYVAREN1    CLEAR THE VAR ENTRY
 LA   2,MYARGLST
 USING AXRARGLST,2
 MVC AXRARGLSTID,MYAXRARGLSTACRO
 LA   5,AXRARGLSTCURVER
 ST   5,AXRARGLSTVER       INITIALIZE THE VERSION
 LA   5,KNUMARGS           OBTAIN THE NUMBER OF ARGUMENTS
 STH   5,AXRARGLSTNUMBER   STORE THE NUMBER OF ARGUMENTS
 DROP 2

LA   5,AXRARGLSTCURVER
ST   5,AXRARGLSTVER       INITIALIZE THE VERSION
```

```
USING AXRARGENTRY,2
        LA  2,MYARGEN1          ADDRESSABILITY TO FIRST ARG ENTRY
        LA  5,KMASTER
        ST  5,AXRARGADDRLOW      STORE ADDRESS OF JOBNAME (*MASTER*)
        OI  AXRARGINPUTFLGS1,AXRARGINPUT INDICATE INPUT ARG
        LA  5,L'KMASTER          OBTAIN LENGTH OF ARG
        ST  5,AXRARGLENGTH       STORE LENGTH OF ARG IN ENTRY
        MVI AXRARGTYPE,AXRARGTYPECHAR  STORE TYPE OF ARG
        DROP 2
        LA  2,MYVARLST
       USING AXRARGLST,2
        MVC AXRARGLSTID,MYAXRVARLSTACRO
        LA  5,AXRARGLSTCURVER
LA  5,KNUMVARS           OBTAIN THE NUMBER OF VARIABLES
STH  5,AXRARGLSTNUMBER   STORE THE NUMBER OF VARIABLES
DROP 2
USING AXRARGENTRY,2
LA  2,MYVAREN1           ADDRESSABILITY TO 1ST VAR ENTRY
LA  5,OUTASTE@
ST  5,AXRARGADDRLOW      STORE OUTPUT ARGUMENT
LA  5,OUTARGNAME
ST  5,AXRARGNAMEADDRLOW STORE ADDRESS OF NAME OF OUTPUT VAR
MVI AXRARGNAMELENGTH,L'OUTARGNAME
OI  AXRARGINPUTFLGS1,AXRARGOUTPUT   INDICATE OUTPUT VAR
MVI AXRARGTYPE,AXRARGTYPEHEXSTRING     INDICATE HEX STRING
LA  5,L'OUTASTE@         OBTAIN LENGTH (IN BYTES)
SLL 5,1       MULT BY 2 - LENGTH IS IN HEX DIGITS (NOT BYTES)
ST  5,AXRARGLENGTH       STORE LENGTH IN VAR ENTRY
DROP 2
AXREXX REQUEST=EXECUTE,NAME=KEXECNAME,REXXARGS=MYARGLST,       *
      REXXVARS=MYVARLST,REXXDIAG=MYAXRDIAG,                    *
      REXXOUTDSN=KOUTDSN
```

```
LTR    15,15
JNZ    FAILLABEL
USING AXRDIAG,2
LA     2,MYAXRDIAG
TM     AXRDIAGFLGS1,AXRDIAGNOEXECRETCODE
JNZ    FAILLABEL
 L      15,AXRDIAGEXECRETCODE
```

```
         LTR   15,15
         JNZ   FAILLABEL
*        EVERYTHING LOOKS GOOD.  PROCESS OUTASTE@ HERE
FAILLABEL DS   0H
* PERFORM ERROR CHECKING
* OUTASTE@ SHOULD CONTAIN MASTER'S ASTE ADDRESS
         PR
KOUTDSN  DC   CL44'RONN.REXXLOG'
KNUMARGS EQU  1
KNUMVARS EQU  1
         DS  0D
MYAXRARGLSTACRO DC C'ARGL'
MYAXRVARLSTACRO DC AL4(AXRVARLSTACRO)
KEXECNAME DC CL8'GETASTE '
KMASTER  DC CL8'*MASTER*'
OUTARGNAME DC CL8'OUTASTE@'
```

```
MYARGLST DS CL(AXRARGLST_LEN)
MYARGEN1 DS CL(AXRARGENTRY_LEN)
MYVARLST DS CL(AXRARGLST_LEN)
MYVAREN1 DS CL(AXRARGENTRY_LEN)
MYAXRDIAG DS CL(AXRDIAG_LEN)
OUTASTE@ DS A
         AXRZARG DSECT=YES,AXRARGLST=YES,AXRARGENTRY=YES,AXRDIAG=YES
         END
```

Common Errors

- **Failure to correctly set up the REXX Run Time Library or the Alternate Library**
  - ***Causes failure to run compiled REXX execs***
  - ***Return/reason code 0C/xxxx0C08, message IRX0157E***
- **Performance issues**
  - ***Service class of AXR or AXRnn address spaces not set up properly (Discretionary – should be privileged)***
  - ***Parent task monitoring exec has much lower priority than subtask running exec (priority inversion)***
    - **Causes failure to apply time limit or Cancel execs**

Best Practices

1 of 4

- **Execs should free any obtained resources**
  - ***Since AXREXX is an authorized API, a level of trust is assumed since the REXX environments are reused unless the request fails (AXREXX non-zero retcode)***
  - ***For TSO=YES requests, SYSREXX detects data sets left allocated by the exec and deallocates them***
    - **Negative performance impact**
  - ***Use Signal Handles: NOVALUE, SYNTAX, HALT***
    - **Return the value of SIGL and the return code to the AXREXX invoker**
- **Provide a consistent diagnostic strategy**
  - ***Pass back return codes and reason codes of the failing API to the AXREXX invoker and any relevant messages***

Best Practices

2 of 4

- **Always specify the DIAG parm on AXREXX invocations**
  - ***Provides valuable diagnostic info***
- **Provide an argument to the exec to run with some level of REXX tracing active**
  - ***Specify RexxOutDsn or CONSDATA on AXREXX invocation to provide a destination for the trace output***

Best Practices

3 of 4

- Create a REXXLIB concatenation and do NOT modify SYS1.SAXREXEC
  - Ensure your exec names do not start with the letters A-I
- Use the TSO=NO environment if possible
  - Benefits – Use of a shared address space
  - Downside – NO TSO Host commands, NO SYSCALL (POSIX) Host commands.
    - Single point of failure – a serious problem could cause a disruption to the AXREXX service

Best Practices

4 of 4

- Coding conventions
  - *Use PROCEDURE/EXPOSE to ensure that changes made by a procedure only affect variables introduced within the procedure*
  - *Prefix global variables with a lower case g and constants with a k. Input and Ouput Args/Vars can be prefixed with In, Out or INOUT.*
    - Variable naming conventions critical to program understanding
- See: http://www.rexxinfo.org/html/rexxinfo1.html#Rexx-Best-Practices
  - *I found Howard Fosdick's article to be helpful (he also has a book which is also informative)*

Possible Future Enhancements

- Use the TMP for TSO=YES in place of the TSO Dynamic Environment
  - *Enables use of the CONSOLE Host Command Environment*
- Enhance AXRWTO/AXRMLWTO
- Allow TSO=NO execs to be run as an operator command
- Operator command to cancel an exec

Possible Future Enhancements

- **New parmlib options**
  - ***Customization of default TIMEINT value (current default is 30 sec)***
  - ***Customization of the number of TSO=NO worker tasks and TSO=YES TSO Server address spaces***
- **Support for updating parmlib options without having to restart SYSREXX**

Session Summary

- System REXX provides a gateway for new style applications to interface with z/OS components
- While the New Face matures, System REXX adds real value today through exploitation possibilities for operator assists and also certain system management processes
- We are keenly interested in your opinions and suggestions that will help to extend and mature this component

➢Any Questions…?

Appendix

- Publication References
- SYSREXX Command Details

Publication References

- **MVS Initialization & Tuning Reference**
  - ***SA22-7592***
- **MVS System Commands**
  - ***SA22-7627***
- **Authorized Assembler Services Guide**
  - ***SA22-7605***
- **Authorized Assembler Services Reference Volume 1**
  - ***SA22-7609***
- **MVS System Codes**
  - ***SA22-7626***
- **MVS Diagnosis: Tools and Service Aids**
  - ***GA22-7589***
- **MVS System Messages Volume 3**
  - ***SA22-7633***

SYSREXX Command

1 of 2

```
@SYSREXX STATUS    OR    @SR ST

AXR0200I SYSREXX STATUS DISPLAY
SYSTEM REXX STARTED AT 11.00.30 ON 09/18/2006
PARMLIB MEMBERS:      AXR00
CPF:  @ (SYSTEM)      AXRUSER:  MEGA
TIMEINT:              30
SUBSYSTEM:            AXR
REQUESTS QUEUED:      0  ACCEPTING NEW WORK
REXX WORKER TASKS:    ACTIVE:   0     TOTAL:   4
                                          IDLE:     4     MAX:     64

                                          ASYNC:    0     SYNC:     0

                                          UNTIMED:  0

TSO SERVER SPACES:    ACTIVE:   0     TOTAL:   0
                                          IDLE:     0     MAX:      8

                                          ASYNC:    0     SYNC:     0

                                          UNTIMED:  0
```

The STATUS sub-function results in a summary display of System REXX environmental information:

- time and date the AXR address space was started
- parmlib member/s used at startup
- command prefix used to identify console initiated REXX execs
- scope of the command prefix
- special userid against which SYSREXX requests may be invoked
- default timeout interval
- number of queued REXX execs requests
- REXX worker tasks available to process REXX execs
  - *Number of tasks actively processing requests*
  - *Number of idle tasks*
  - *Total number of tasks (active+idle)*
  - *Maximum number of tasks that may be started*
  - *Number of synchronous and asynchronous requests*
  - *Number of untimed requests*
- TSO server address spaces available to process REXX execs
  - *Number of address spaces actively processing requests*
  - *Number of idle address spaces*
  - *Total number of address spaces (active+idle)*
  - *Maximum number of address spaces that may be started*
  - *Number of synchronous and asynchronous requests*
  - *Number of untimed requests*

SYSREXX Command

2 of 2

@SYSREXX STATUS,**DETAIL** OR @SR ST,D

```
AXR0201I SYSREXX STATUS DETAIL
EXEC=WAITLOOP CJBN=AXR      CASID=0015 TSO=Y T/L=00.00.30
  REQTOKEN=0000520000000000BF3A704A6511A3B5
  EJBN=AXR02    EASID=0033 TCB=006FF098 CPU=000.004S TIME=005.739S
EXEC=INFINITE CJBN=AXR      CASID=0015 TSO=Y T/L=00.00.30
  REQTOKEN=0000540000000000BF3A704C2088405C
  EJBN=AXR03    EASID=0032 TCB=006FF098 CPU=000.006S TIME=003.925S
```

The DETAIL parameter causes detailed information about executing REXX execs to be displayed:

- name of the REXX exec
- requester Jobname and ASID
- type of request - TSO=N or TSO=Y
- time limit for the request
- unique request token associated with the request
- execution Jobname, ASID and TCB address for TSO=Y requests
- cpu seconds currently used
- current elapse time seconds