



IBM Systems

SDSF REXX

IBM Poughkeepsie

Trademarks

The following are trademarks of the International Business Machines Corporation in the United States and/or other countries.

**IBM®
MVS
JES2
JES3
RACF®
REXX
z/OS®
zSeries®**

* Registered trademarks of IBM Corporation

The following are trademarks or registered trademarks of other companies.

Java and all Java-related trademarks and logos are trademarks of Sun Microsystems, Inc., in the United States and other countries.

Linux is a registered trademark of Linus Torvalds in the United States, other countries, or both.

Microsoft, Windows and Windows NT are registered trademarks of Microsoft Corporation.

UNIX is a registered trademark of The Open Group in the United States and other countries.

SET and Secure Electronic Transaction are trademarks owned by SET Secure Electronic Transaction LLC.

* All other products may be trademarks or registered trademarks of their respective companies.

Notes:

Performance is in Internal Throughput Rate (ITR) ratio based on measurements and projections using standard IBM benchmarks in a controlled environment. The actual throughput that any user will experience will vary depending upon considerations such as the amount of multiprogramming in the user's job stream, the I/O configuration, the storage configuration, and the workload processed. Therefore, no assurance can be given that an individual user will achieve throughput improvements equivalent to the performance ratios stated here.

IBM hardware products are manufactured from new parts, or new and serviceable used parts. Regardless, our warranty terms apply.

All customer examples cited or described in this presentation are presented as illustrations of the manner in which some customers have used IBM products and the results they may have achieved. Actual environmental costs and performance characteristics will vary depending on individual customer configurations and conditions.

This publication was produced in the United States. IBM may not offer the products, services or features discussed in this document in other countries, and the information may be subject to change without notice. Consult your local IBM business contact for information on the product or services available in your area.

All statements regarding IBM's future direction and intent are subject to change or withdrawal without notice, and represent goals and objectives only.

Information about non-IBM products is obtained from the manufacturers of those products or their published announcements. IBM has not tested those products and cannot confirm the performance, compatibility, or any other claims related to non-IBM products. Questions on the capabilities of non-IBM products should be addressed to the suppliers of those products.

Prices subject to change without notice. Contact your IBM representative or Business Partner for the most current pricing in your geography.

Overview

- z/OS V1R9 SDSF adds support for the REXX programming language
 - z/OS V1R9 GA September, 2007
- Use REXX to quickly develop scripts to perform complex and repetitive tasks
- Little or no training needed if you already know SDSF
- Simpler and more powerful alternative to SDSF batch
 - Lets you include logic
- Answers or partially answers 14 customer requirements

Overview

- With SDSF's REXX, you can perform most of the tasks that you can perform interactively, such as:
 - Display and modify jobs
 - Display and modify resources and devices
 - Browse SYSOUT data sets
 - Print SYSOUT data sets
- Use the same panel commands, action characters and column overtypes as with interactive SDSF

Getting Started

In a basic SDSF REXX exec, you:

1. Add the REXX host command environment; before issuing any SDSF commands, using **ISFCALLS**
 2. Issue an SDSF command to access a panel, using **ISFEXEC**
 3. Issue an action character or “overtime” a column using **ISFACT**
- Data is returned in stem variables
 - Use new special variables to control results
 - These correspond to SDSF commands such as PREFIX and OWNER

Quick Start Example – Cancel a Job

```
rc=isfcalls("ON")
Address SDSF "ISFEXEC ST"
do ix=1 to JNAME.0 (variable names same as FLD names)
  if pos("KEN",JNAME.ix) = 1 then
    Address SDSF "ISFACT ST TOKEN("TOKEN.ix") PARM(NP P)"
    [...lines omitted...]
rc=isfcalls("OFF")
```

Add host command environment

Access the ST panel

Find the job

Take an action on the job

Remove the host command environment (after closing the loop)

Accessing an SDSF Panel with REXX

- Use ISFEXEC to access a panel
- Syntax:
Address SDSF "ISFEXEC *sdsf-command* (*options*)"

– *sdsf-command* is the same SDSF command as you use interactively, including parameters, for example:

- Address SDSF "ISFEXEC DA"
- Address SDSF "ISFEXEC CK ALL"

DA command

CK command with the ALL parameter

Accessing an SDSF Panel - Options

Options you can use when accessing a panel with ISFEXEC or ISFACT:

- PREFIX: specify a prefix for variables that are created
- ALTERNATE: use the alternate field list
- DELAYED: include delayed-access columns
- NOMODIFY: don't return row tokens for use in modifying values
- VERBOSE: add diagnostic messages to the isfmsg2.stem variable (more on this later)

Accessing an SDSF Panel - Data

- SDSF builds stem variables that correspond to the panel's rows and columns
 - *column-name.index* format
 - *column-name* is the name used on an FLDENT statement (not the column title), for example:
FLDENT COLUMN(**OWNERID**),TITLE(OWNER),WIDTH(8)
 - *index* is the number of the row
 - 0 index is the number of variables in the stem
- Display the column names with the COLSHELP command

Stem Variables for Panel Data - Example

Stem variables and values for columns on the Status panel:

JNAME.0=2

JNAME.1=KENA

JNAME.2=BOBB

OWNERID.0=2

OWNERID.1=KEN

OWNERID.2=BOB

... and so on

Count of job name variables

Job name for row 1

Job name for row 2

Count of owner variables

Special Variables to Control SDSF

- Special variables for use with SDSF REXX
 - Defined by SDSF
 - Some correspond to SDSF commands
 - Others provide access to fields or data, such as the title line on an SDSF panel
 - Names start with “ISF”

Special Variables – Limit Panel Response

- Special variables with panel commands:
 - Limit the response when accessing a panel
 - Use before invoking ISFEXEC or ISFACT

- Examples

isfprefix=*

Corresponds to the command PREFIX *

isfowner=ken

Corresponds to the command OWNER KEN

isffilter=jprio gt 5

Corresponds to the command
FILTER PRTY GT 5

isfcols="JNAME JOBID OWNERID ACTSYS"

Limits the column variables created

Special Variables - Others

- **ISTLINE** – contents of the panel title line
 - **ISFROWS** – number of rows on the panel
 - **ISFDELAY** – delay limit for system command responses (like **SET DELAY** command)
 - **ISFACTIONS** – controls the list of valid action characters (like **SET ACTION** command)
 - **ISFSERVER** – names the SDSF server to be used (like the server name on the SDSF command)
- ... many more

A More Complete REXX Example

```
rc=isfcalls("ON")
Address SDSF "ISFEXEC ST"
if rc<>0 then Exit 20
fixedfield=word(isfcols,1)
say "Number of rows returned" isfrows
do ix=1 to isfrows
  say "Now processing job:" value(fixedField"."ix)
  do jx=1 to words(isfcols)
    col=word(isfcols,jx)
    say " Column" col"."ix "has the value:" value(col"."ix)
  end
end
rc=isfcalls("OFF")
```

Variable **isfcols** contains a list of columns, with fixed field first

Variable **isfrows** contains a count of rows

A More Complete Example – Output from the Exec

Number of rows returned: 35

Now processing job: RJONES

Column JNAME.1 has the value: RJONES

Column TOKEN.1 has the value: 2hg0EwU7o5

Column JOBID.1 has the value: T0000034

Column OWNERID.1 has the value: RJONES

Column JPRIO.1 has the value: 15

Column QUEUE.1 has the value: EXECUTION

... and so on

Now processing job: TCAS

Column JNAME.2 has the value: TCAS

... and so on

Message Variables

- Message variables contain SDSF messages
 - **isfmsg** contains the SDSF short message (displayed in the upper right corner on an SDSF panel)
 - **isfmsg2.** stem contains the SDSF numbered messages
 - **Isfulog** stem is for the user log (ULOG)
- Check after each SDSF request to ensure the request was successful

Message Variables Example with Slash

Address SDSF "ISFEXEC '/\$da' (WAIT"

Issue the w/\$da command with WAIT option

if isfmsg<> "" then

Check for a short message

 Say "isfmsg is:" isfmsg

do ix=1 to isfmsg2.0

Check for a numbered message. The 0 stem contains a count of the numbered messages.

 Say "isfmsg2."ix "is:" isfmsg2.ix

end

do ix=1 to isfulog.0

Check the ULOG

 Say "isfulog."ix "is" isfulog.ix

end

Actions

Should also check the return code from the SDSF command, for example: if rc<>0 then ...

Return codes for ISFEXEC and ISFACT:

- **00** The request completed successfully.
- **08** An incorrect or invalid parameter was specified for an option or command.
- **12** A syntax error occurred parsing a host environment command.
- **16** The user is not authorized to invoke SDSF.
- **20** A request failed due to an environmental error.
- **24** A request failed due to an environmental error.

Actions

- Use the ISFACT command to issue an action character or modify a value (overtyping a column)

- Syntax:

Address SDSF "ISFACT *sdsf-cmd* TOKEN("*token*")
PARM(*parms*) (*options*)"

- *sdsf-cmd* is the same SDSF command you used with ISFEXEC to access the panel
- *token* identifies the row
- *parms* identifies the column and action

Actions - continued

TOKEN('token')

- TOKEN stem is returned by ISFEXEC
- Each row is identified with a TOKEN variable
- TOKEN is used as input to ISFACT

PARM(*parms*)

- Describes the action or modification
- *column value* format, for example:
 - PARM(OCLASS A FORMS 1234)
 - PARM(NP C)

Change both
class & forms

Use NP for action characters

Example - Change Output Forms

```
isfprefix="**"
```

Set filters

```
isfowner="RJONES"
```

Access O panel to set variables

```
Address SDSF "ISFEXEC O"
```

```
do ix=1 to JNAME.0
```

Find a row with job name BOB

```
if pos("BOB",JNAME.ix) = 1 then
```

```
do
```

```
Address SDSF "ISFACT O TOKEN('TOKEN.ix')  
PARM(FORMS 1234)"
```

```
end
```

Use the token for that row to identify it, enclosing it in single quotes

```
end
```

Change forms

Browse Job Data Sets

- Use ISFACT to issue the SA action character against a job
 - Allocates the data set (free=close)
 - REXX only
- Allocated ddname is returned in **isfddname.** stem variable
- Data set name is in **isfdsname.** stem variable
- Use EXECIO to read the data set

Example: Browse Job Data Sets

Address SDSF "ISFEXEC ST"

Access the ST panel, then use logic to find a job (not shown)

...

Address SDSF "ISFACT ST TOKEN("TOKEN.ix") PARM(NP SA)"

Issue SA action

do jx=1 to isfddname.0

Loop through ddnames

Say "Now reading" isfdsname.jx

"EXECIO * DISKR" isfddname.jx "(STEM line. FINIS"

Say "Lines read" line.0

EXECIO reads the data set

do kx=1 to line.0

Say " line."kx "is:" line.kx

end

end

Browse Health Check Output

- Use ISFEXEC to access the CK panel
- Use ISFACT to issue the S action character to browse check output
 - Output is placed in the **isfline.** stem special variable

Printing a Job Data Set

- Print a job data set
 - Use X, XS, XD action characters
 - ISFACT PARM(NP X)
- Attributes of print data set are controlled through special variables, for example:
 - isfpqrtclass="U"
 - isfpqrtcopies="2"
 - isfpqrtdest="ken"
 - isfpqrtformdef="ffff"
 - isfpqrtforms="8888"
 - isfpqrtpagedef="pppp"
 - isfpqrtprmode="pmode"

Avoiding Duplicate Variable Names

Use the **PREFIX** option on ISFEXEC and ISFACT to add a prefix to variable names created by SDSF

- Prevents duplicate variable names in existing scripts
 - Needed when accessing the job data set panel, so that column variables don't conflict
- Format: (PREFIX *prefix*)

Example: Using the PREFIX Option

```
Address SDSF "ISFACT ST TOKEN("TOKEN.ix") PARM(NP '?')  
  (PREFIX jds_)"
```

Access JDS using NP ? and define a prefix for all JDS variables.

```
do jx=1 to jds_DDNAME.0  
  say "DSName is" jds_DSNAME.jx  
  Say "Stepname is" jds_STEPN.jx  
  Say "Procstep is" jds_PROCS.jx  
end
```

References to variables all include the prefix

Security

- SDSF security applies to using SDSF through REXX
- No changes to ISFPARMS or SAF
- IBM recommends SAF for security instead of ISFPARMS for better control and auditing

Security – Assigning a User to a Group

SDSF assigns users to a group in ISFPARMS with:

- SAF: checks resource
GROUP.*group-name.server-name* in the SDSF class
- ISFPARMS: Uses user ID, logon proc, etc.
 - With REXX, special values are assigned as follows:
 - Logon proc name: Set to REXX
 - TSO authority: Set to JCL authority
 - Terminal name: Derived from SAF or TSO based on the current environment

Diagnosing Problems

- Check ISFMSG variables and ISFMSG2. stem variable
- Use the VERBOSE option on ISFEXEC and ISFACT
 - Issues a message (in the isfmsg2 stem variable) for each variable that is set
 - Useful in diagnosing problems such as ‘why doesn’t my job name comparison work?’
 - Example: Address SDSF “ISFEXEC DA (VERBOSE)”

Results:

ISF146I REXX variable JOBID.1 set, return code 00000001 value is ‘J0000040’.

ISF146I REXX variable OWNERID.1 set, return code 00000001 value is ‘RJONES’.

Diagnosing Problems (cont.)

- ISFDIAG variable
 - Intended for use by IBM Service
 - Contains internal reason codes for each request
 - You may be asked to employ it if you call IBM with a problem

References

- Issue the **REXXHELP** command while using SDSF under ISPF
- See *SDSF Operation and Customization*:
<http://publibz.boulder.ibm.com/epubs/pdf/isf4cs70.pdf>
- SDSF Web page, which will include examples for use with ISPF's MODEL command:
<http://www.ibm.com/servers/eserver/zseries/zos/sdsf/>
- New Redbook!
 - Loaded with interesting examples and experiences

SDSF REXX Redbook

Title: *Implementing REXX Support in SDSF*,
SG24-7419-00

<http://w3.itso.ibm.com/abstracts/sg247419.html?Open>

Abstract:

This IBM Redbooks publication describes the new support and provides sample REXX execs that exploit the new function and that perform real-world tasks related to operations, systems programming, system administration, and automation.

SDSF REXX Redbook - Topics

Chapter 1. Issuing a system command

Chapter 2. Copying SYSOUT to a PDS

Chapter 3. Bulk job update processor

Chapter 4. SDSF support for the COBOL language

Chapter 5. Searching for a message in SYSLOG

Chapter 6. Viewing SYSLOG

Chapter 7. Reviewing execution of a job

Chapter 8. Remote control from other systems

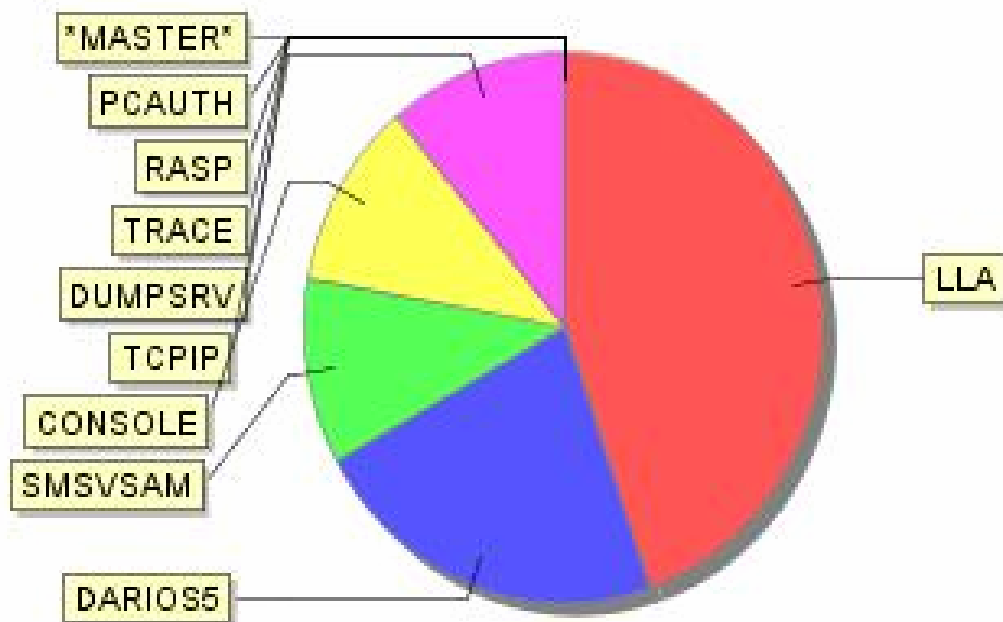
Chapter 9. JOB schedule and control

Chapter 10. SDSF data in graphics

Chapter 11. Extended uses

SDSF REXX Redbook - Examples

CPU % . Top 10 consumers



- LLA ● DARIOS5 ● SMSVSAM ● CONSOLE ● TCPIP ● *MASTER* ● PCAUTH ● RASP
- TRACE ● DUMPSRV

SDSF REXX Redbook - Examples

