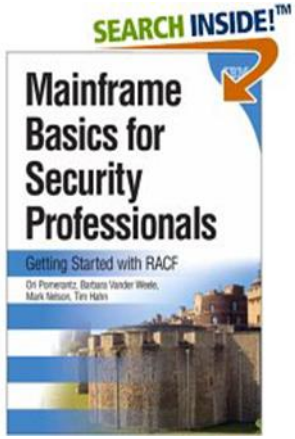
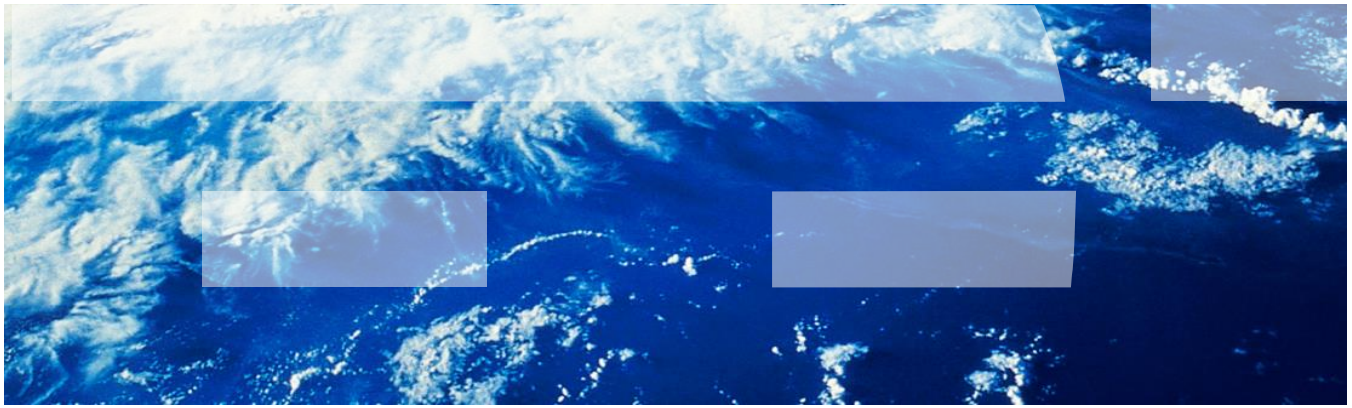
The IBM® Health Checker for z/OS®, IRRXUTIL, and System REXX: A Triumphant Trio!

Session 19646

Mark Nelson, CISSP®, CSSLP®
RACF Design and Development
IBM Poughkeepsie
markan@us.ibm.com





Trademarks

IBM, the IBM logo, and ibm.com are trademarks or registered trademarks of International Business Machines Corporation in the United States, other countries, or both. If these and other IBM trademarked terms are marked on their first occurrence in this information with a trademark symbol (® or ™), these symbols indicate U.S. registered or common law trademarks owned by IBM at the time this information was published. Such trademarks may also be registered or common law trademarks in other countries. A current list of IBM trademarks is available on the Web at "Copyright and trademark information" at www.ibm.com/legal/copytrade.shtml

Java and all Java-based trademarks are trademarks of Sun Microsystems, Inc. in the United States, other countries, or both.

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

Other company, product, or service names may be trademarks or service marks of others.

Agenda

- **The IBM Health Checker for z/OS**
  - History of the IBM Health Checker for z/OS
  - Structure
  - The Health Check
  - Check “Philosophy”
  - The RACF Health Checks
  - Check Output

- **An Introduction to IRRXUTIL**
  - What is IRRXUTIL?
  - Relationship to the R_admin application programming interface
  - Authorization for R_admin
  - Invocation syntax
  - Sample invocation
  - Where to find field names
  - Considerations
  - Return codes
  - Returned data
  - Samples

- **Writing a System REXX Health Check which Uses IRRXUTIL**

An Introduction to the IBM Health Checker for z/OS

The IBM Health Checker for z/OS

- **What is the IBM Health Checker for z/OS?**
  - **Originally a tool developed by IBM International Technical Support Organization (ITSO) to address common configuration and setup errors**
    - 15-20% of system outages attributed to setup and configuration
    - Implemented as a batch job, with 37 checks in 2003
    - Delivered as a web download
  - **Ever since z/OS V1.7, the IBM Health Checker for z/OS was integrated into z/OS**
    - Implemented as a started task
    - Initially, 55 checks shipped with z/OS… with z/OS V2.2 160+ checks!
    - IBM checks are shipped with components
    - Application programming interfaces (APIs) for check management
    - Extensive SDSF support

Structure of the IBM Health Checker for z/OS

- **The IBM Health Checker for z/OS consists of:**
  - A managing address space (the “backbone”)
  - A utility (HZSPRINT) for collecting check output
  - The Health Checks, which can be written by:
    - Individual IBM components (such as RACF, UNIX® System Services)
    - ISVs
    - You!
    - …. And starting with z/OS V1.9, you can write the check in System REXX

- **A check is identified by a:**
  - 1-32 character check name, such as:
    - CSV_APF_EXISTS
    - GRS_CONVERT_RESERVES
    - RACF_IBMUSER_REVOKED
  - 1-16 character check owner
    - The owner for an IBM-supplied check begins with IBM, for example:
      IBMCSV, IBMGRS, and IBMRACF

- **Checks can execute in:**
  - The Health Checker for z/OS address space (“Local check”)
  - Another address space (“Remote Check”)
    - System REXX checks are a type of remote check as they execute in a System REXX address space

Check “Philosophy”

- **Health Checks raise exceptions and make recommendations, ***but they do not automatically take any actions***
  - You must review the recommendation and ensure that it is appropriate for your environment
- **When an exception is found, Health Checks present the entire message information, including the “explanation”, “systems programmer response”, etc., along with pointers to relevant documentation.**
- **Checks which find no exception clearly state that no exception was found.**
- **Checks which are not applicable to the current environment place themselves in a “not applicable” status and will not run unless triggered.**

The Health Check…

- **Associated with each check is information about its execution:**
  - Execution state:
    - ACTIVE or INACTIVE
  - How often the check runs
    - ONETIME, hh:mm
  - The severity of the check, which influences how check output is issued
    - HIGH, MEDIUM, LOW, NONE
  - WTOTYPE
    - CRITICAL, EVENTUAL, INFORMATIONAL, HARDCOPY, NONE

- **Some checks accept parameters which direct the processing of the check or set thresholds**

- **Check information is set by the check writer, but can be changed by the installation by:**
  - Policy statements in the HZSPRMxx member of PARMLIB
  - MVS MODIFY Command (F HC ….)
- **Starting with z/OS V2R1, the IBM Health Checker for z/OS starts automatically**

The RACF Health Checks

- **RACF ships these Health Checks:**
  - RACF_AIM_STAGE
  - RACF_BATCHALLRACF*
  - RACF_CERTIFICATE_EXPIRATION
  - RACF_ENCRYPTION_ALGORITH
  - RACF_GRS_RNL
  - RACF_IBMUSER_REVOKED
  - RACF_ICHAUTAB_NONLPA
  - RACF_PASSWORD_CONTROLS
  - RACF_RRSF_RESOURCES
  - RACF_SENSITIVE_RESOURCES
  - RACF_UNIX_ID
  - RACF_<class-name>_ACTIVE
    - Verifies that the class <class-name> is active
    - Check is performed for CSFKEY, CSFSERV, FACILITY, JESJOBS*, JESSPOOL*, OPERCMDS, TAPEVOL, TEMPDSN, TSOAUTH, UNIXPRIV
  - Installation-defined RACF Health Checks

* With OA48716, which has PTFs for z/OS V1.13, V2.1, and V2.2

Check Output

- **The output of a check consists of:**
  - Write to Operator messages (WTO)s, which are written with the routing codes and descriptor codes associated with the check
  - Messages written to the Health Check message buffer, which can be:
    - Kept in storage (most recent check invocation only)
    - Written to a log stream

- **Check output can be processed with:**
    - SDSF, using the “CK” panels
    - Using the HZSPRINT utility

Updated SDSF Primary Option Panel

```
HQX7720 -----------------  SDSF PRIMARY OPTION MENU  -------------------------

 DA     Active users                      INIT   Initiators
 I      Input queue                       PR     Printers
 O      Output queue                      PUN    Punches
 H      Held output queue                 RDR    Readers
 ST     Status of jobs                    LINE   Lines
                                          NODE   Nodes
 LOG    System log                        SO     Spool offload
 SR     System requests                   SP     Spool volumes
 MAS    Members in the MAS
 JC     Job classes                       RM     Resource monitor
 SE     Scheduling environments           CK     Health checker
 RES    WLM resources

 Licensed Materials - Property of IBM

 5694-A01 (C) Copyright IBM Corp. 1981, 2005. All rights reserved.
 US Government Users Restricted Rights - Use, duplication or
 COMMAND INPUT ===> ck                                   SCROLL ===> PAGE
  F1=HELP      F2=SPLIT     F3=END       F4=RETURN    F5=IFIND     F6=BOOK
  F7=UP        F8=DOWN      F9=SWAP     F10=LEFT     F11=RIGHT    F12=RETRIEVE
```

SDSF Check Selection Panel

```
SDSF HEALTH CHECKER DISPLAY  RACFR21                    LINE 55-69 (143)
COMMAND INPUT ===>                                        SCROLL ===> HALF
PREFIX=*  DEST=(ALL)  OWNER=*  SORT=NAME/A  SYSNAME=  FILTERS=1
NP    NAME                              CheckOwner       State              Status
      PDSE_SMSPDSE1                     IBMPDSE          ACTIVE(ENABLED)    EXCEPT
      RACF_AIM_STAGE                    IBMRACF          ACTIVE(ENABLED)    SUCCES
      RACF_CERTIFICATE_EXPIRATION       IBMRACF          ACTIVE(ENABLED)    EXCEPT
      RACF_CSFKEYS_ACTIVE               IBMRACF          ACTIVE(ENABLED)    EXCEPT
      RACF_CSFSERV_ACTIVE               IBMRACF          ACTIVE(ENABLED)    EXCEPT
      RACF_ENCRYPTION_ALGORITHM         IBMRACF          ACTIVE(ENABLED)    EXCEPT
      RACF_FACILITY_ACTIVE              IBMRACF          ACTIVE(ENABLED)    SUCCES
      RACF_GRS_RNL                      IBMRACF          ACTIVE(DISABLED)   ENV N/
      RACF_IBMUSER_REVOKED              IBMRACF          ACTIVE(ENABLED)    EXCEPT
      RACF_ICHAUTAB_NONLPA              IBMRACF          ACTIVE(ENABLED)    SUCCES
      RACF_OPERCMDS_ACTIVE              IBMRACF          ACTIVE(ENABLED)    SUCCES
      RACF_PASSWORD_CONTROLS            IBMRACF          ACTIVE(ENABLED)    EXCEPT
 S    RACF_SENSITIVE_RESOURCES          IBMRACF          ACTIVE(ENABLED)    EXCEPT
      RACF_TAPEVOL_ACTIVE               IBMRACF          ACTIVE(ENABLED)    EXCEPT
      RACF_TEMPDSN_ACTIVE               IBMRACF          ACTIVE(ENABLED)    EXCEPT
      RACF_TSOAUTH_ACTIVE               IBMRACF          ACTIVE(ENABLED)    SUCCES
      RACF_UNIX_ID                      IBMRACF          ACTIVE(ENABLED)    EXCEPT
      RACF_UNIXPRIV_ACTIVE              IBMRACF          ACTIVE(ENABLED)    EXCEPT
      RCF_PCCA_ABOVE_16M                IBMRCF           ACTIVE(ENABLED)    SUCCES
PF 1=HELP      2=SPLIT     3=END       4=RETURN    5=IFIND     6=BOOK
PF 7=UP        8=DOWN      9=SWAP     10=LEFT     11=RIGHT    12=RETRIEVE
```

SDSF Browse Check Output Panel

```
CHECK(IBMRACF,RACF_SENSITIVE_RESOURCES)
SYSPLEX:    LOCAL     SYSTEM: RACFR22
START TIME: 06/07/2016 22:53:39.643246
CHECK DATE: 20120106  CHECK SEVERITY: HIGH

                          APF Dataset Report

S Data Set Name                              Vol    UACC Warn ID*  User
- ------------------------------------------ ------ ---- ---- ---- ----
E ASM.SASMMOD1                               ZDR22B Read Yes  ****
V ATC.V2R1M4.AUTHLIB                         DRVPSL
V CBC.SCBCCMP                                ZDR22B
  CBC.SCCNCMP                                ZDR22B Read No   ****
  CBC.SCLBDLL                                ZDR22B Read No   ****
  CBC.SCLBDLL2                               ZDR22B Read No   ****
  CEE.SCEERUN                                ZDR22B Read No   ****
  CEE.SCEERUN2                               ZDR22B Read No   ****
  F1=HELP      F2=SPLIT     F3=END       F4=RETURN    F5=IFIND     F6=BOOK
  F7=UP        F8=DOWN      F9=SWAP     F10=LEFT     F11=RIGHT    F12=RETRIEVE
```

SDSF Browse Check Output Panel …

```
 SDSF OUTPUT DISPLAY RACF_SENSITIVE_RESOURCES      LINE 87      COLUMNS 02- 81
 COMMAND INPUT ===>                                            SCROLL ===> PAGE
                          RACF Dataset Report

S Data Set Name                              Vol    UACC Warn ID*  User
- ------------------------------------------ ------ ---- ---- ---- ----
  RACFDRVR.RACF317                           RDB317 None No   ****

* High Severity Exception *

RACS204E The RACF_SENSITIVE_RESOURCES check has found one or
more potential errors in the security controls on this system.

  Explanation:  The RACF security configuration check has found one or
    more potential errors with the system protection mechanisms.

  System Action:  The check continues processing. There is no effect on
    the system.

  Operator Response:  Report this problem to the system security administrator and the
     and the system auditor.
```

SDSF Browse Check Output Panel …

```
SDSF OUTPUT DISPLAY RACF_SENSITIVE_RESOURCES      LINE 105     COLUMNS 02- 81
 COMMAND INPUT ===>                                            SCROLL ===> PAGE

  System Programmer Response:  Examine the report that was produced by
    the RACF check. Any data set which has an "E" in the "S" (Status)
    column has excessive authority allowed to the data set. That
    authority may come from a universal access (UACC) or ID(*) access
    list entry which is too permissive, or if the profile is in WARNING
    mode. If there is no profile, then PROTECTALL(FAIL) is not in
    effect. Any data set which has a "V" in the "S" (Status) field is
    not on the indicated volume. Remove these data sets from the list
    or allocate the data sets on the volume. Any data set which has an
    "M" in the "S" (Status) field has been migrated.

    The APF_LIBS check provides additional analysis of the non-RACF
    aspects of your APF list.

    If the "S" field contains an "E" or is blank, then blanks in the
    UACC, WARN, and ID(*) columns indicate that there is no RACF
  F1=HELP       F2=SPLIT      F3=END        F4=RETURN     F5=IFIND      F6=BOOK
  F7=UP         F8=DOWN       F9=SWAP      F10=LEFT      F11=RIGHT     F12=RETRIEVE
```

SDSF Browse Check Output Panel …

```
 Display  Filter  View  Print  Options  Help
-------------------------------------------------------------------------------
SDSF OUTPUT DISPLAY RACF_SENSITIVE_RESOURCES      LINE 120     COLUMNS 02- 81
COMMAND INPUT ===>                                            SCROLL ===> PAGE

   If the "S" field contains an "E" or is blank, then blanks in the
   UACC, WARN, and ID(*) columns indicate that there is no RACF
   profile protecting the data set. Data sets which do not have a RACF
   profile are flagged as exceptions, unless SETROPTS PROTECTALL(FAIL)
   is in effect for the system.

   If a valid user ID was specified as a parameter to the check, that
   user's authority to the data set is checked. If the user has an
   excessive authority to the data set, that is indicated in the USER
   column. For example, if the user has ALTER authority to an
   APF-authorized data set, the USER column contains ">Read" to
   indicate that the user has more than READ authority to the data set.

 Problem Determination:  See the RACF System Programmer's Guide and
   the RACF Auditor's Guide for information on the proper controls for
   your system.

 F1=HELP      F2=SPLIT     F3=END       F4=RETURN    F5=IFIND     F6=BOOK
 F7=UP        F8=DOWN      F9=SWAP     F10=LEFT     F11=RIGHT    F12=RETRIEVE
```

SDSF Browse Check Output Panel …

```
SDSF OUTPUT DISPLAY RACF_SENSITIVE_RESOURCES      LINE 138      COLUMNS 02- 81
 COMMAND INPUT ===>                                             SCROLL ===> PAGE
  Source:
    RACF System Programmer's Guide
    RACF Auditor's Guide

  Reference Documentation:
    RACF System Programmer's Guide
    RACF Auditor's Guide

  Automation:  None.

  Check Reason:  Sensitive resources should be protected.

 END TIME: 06/07/2016 22:53:55.529345  STATUS: EXCEPTION-HIGH
******************************** BOTTOM OF DATA ********************************



  F1=HELP       F2=SPLIT      F3=END        F4=RETURN     F5=IFIND      F6=BOOK
  F7=UP         F8=DOWN       F9=SWAP      F10=LEFT      F11=RIGHT     F12=RETRIEVE
```

z/OS Console Messages from Health Checks

```
   *RACFR17 *HZS0015E PROBLEM WITH HZSPDATA DATA SET:
    *DD NOT DEFINED
    *RACFR17 *10 HZS0013A SPECIFY THE NAME OF AN EMPTY HZSPDATA DATA SET
     $HASP003         SPECIFICATION
     RACFR17  $HASP646 12.0000 PERCENT SPOOL UTILIZATION
     RACFR17  HZS0001I CHECK(IBMCSV,CSV_APF_EXISTS):
     CSVH0957E Some problem(s) were found with data set(s) in the APF list.
    *RACFR17 *HZS0003E CHECK(IBMRACF,RACF_SENSITIVE_RESOURCES):
    *IRRH204E The RACF_SENSITIVE_RESOURCES check has found one or
    *more potential errors in the security controls on this system.
00   RACFR17  $HASP003 RC=(52),                                          C
     $HASP003 RC=(52),S1-999  - NO SELECTABLE ENTRIES FOUND MATCHING
     $HASP003         SPECIFICATION
     RACFR17  $HASP003 RC=(52),                                          C
     $HASP003 RC=(52),T1-999  - NO SELECTABLE ENTRIES FOUND MATCHING
     $HASP003         SPECIFICATION
     RACFR17  $HASP650 Q,Q=W     INVALID OPERAND OR MISPLACED OPERAND
     RACFR17  $HASP893 VOLUME(SPOOL1)                                    C
     $HASP893 VOLUME(SPOOL1)  STATUS=ACTIVE,SYSAFF=(ANY),TGNUM=175,
     $HASP893                 TGINUSE=21,TRKPERTGB=3,PERCENT=12
     RACFR17  $HASP646 12.0000 PERCENT SPOOL UTILIZATION
 IEE612I CN=C3E0S17  DEVNUM=03E0 SYS=RACFR17
```

An Introduction to IRRXUTIL

What is IRRXUTIL?

- **IRRXUTIL allows a REXX program to extract RACF profile and SETROPTS data**
  - Supports the extraction of USER, GROUP, CONNECT, GENERAL RESOURCE and SETROPTS data from RACF
  - Data set extraction not supported
  - Digital Certificate information not supported
- **IRRXUTIL places the returned data directly into REXX variables which can then be easily used simply by referencing the REXX variables**
- **IRRXUTIL uses the R_admin callable service to extract data**

What is the R_admin Callable Service?

- **The R_admin callable service (IRRSEQ00) is an assembler programming interface which allows for management of RACF profiles and system wide settings (SETROPTS)**

- **R_admin allows you to:**

  – Execute RACF commands
    - With the exception of RVARY, BLKUPD, RACLINK, RACF operator commands (TARGET, SET, SIGNOFF, etc.)

  – Update/Extract profile information into a tokenized format
    - USER, GROUP, user-to-group connections, general resources including access lists
    - Data set profiles (UPDATE only)

  – Set/Extract SETROPTS information
    - SMF Unload-like format
    - “Tokenized” format

  … and more!

Authorization for R_admin

- **R_admin may be invoked by authorized and unauthorized callers.**
  - Authorization is required to set or change the user ID under which the function is performed.
  - Non-authorized callers cannot use the R_admin update function codes
  - Non-authorized callers must have READ authority to a function-specific resource in the FACILITY class. For example:
    - IRR.RADMIN.command for a RACF command (such as IRR.RADMIN.LISTUSER for an LU command)
    - IRR.RADMIN.SETROPTS.LIST to extract SETROPTS data
- **<u>You must authorize IRRXUTIL users to:</u>**
  - The R_admin service
  - The underlying profile / SETROPTS information

IRRXUTIL Invocation Syntax

- **myrc=**IRRXUTIL(*function,type,profile,stem,prefix,generic*)
  - ***Function:*** “EXTRACT” or “EXTRACTN”
    - EXTRACT: Get the information for the name profile
    - EXTRACTN: Get the information for the next profile
  - ***Type:*** “USER”, “GROUP”, “CONNECT”, “_SETROPTS”, *general resource class.* DATASET not supported.
  - ***Profile:*** Profile to extract. Case sensitive. Specify '_SETROPTS' for SETROPTS data.
  - ***Stem:*** REXX stem variable name to populate with results. Do not put the '.' at the end. Prevents collisions with other variables in the program.
  - ***Prefix:*** Optional prefix for returned variable name parts
  - ***Generic:*** Optional, 'TRUE' or 'FALSE' (uppercase). Applies to general resource profiles only.

A Quick Example

- **Here is a simple program which retrieves a general resource profile and dumps the access list.**

```
/* REXX */
myrc=IRRXUTIL("EXTRACT","FACILITY","BPX.DAEMON","RACF","","FALSE")
say "Owner: "RACF.BASE.OWNER.1
Say "ACL:"
do a=1 to RACF.BASE.ACLCNT.REPEATCOUNT
   Say "  "||RACF.BASE.ACLID.a||":"||RACF.BASE.ACLACS.a
end
```

```
READY
EX 'SAMPLE.CLIST(IRREXXRS)'
Owner: IBMUSER
ACL:
      IBMUSER:READ
      WEBSRVR:READ
      MEGA:READ
      LDAPSRVR:READ
      FTPD:READ
READY
```

- Note the complete lack of parsing code. Just retrieve the profile and directly access the required data.
- Note also the lack of return code checking. Bad code. No donut!

Where Do You Find Field Names?

- **z/OS Security Server RACF Callable Services contains tables which document every segment and field name supported by R_admin in appendix A.2**
- **Fields which are 'Returned on Extract Requests' are supported by IRRXUTIL.**

Table 107 TSO segment fields

Field name	Flag byte values	ADDUSER/ALTUSER keyword reference	Allowed on add requests	Allowed on alter requests	Returned on extract requests
ACCTNUM	'Y'	TSO(ACCTNUM (xx))	Yes	Yes	Yes
	'N'	TSO (NOACCTNUM)	No	Yes	
DEST	'Y'	TSO (DEST (xx))			
	'N'	TSO(NODEST)			

Table 121 BASE segment fields

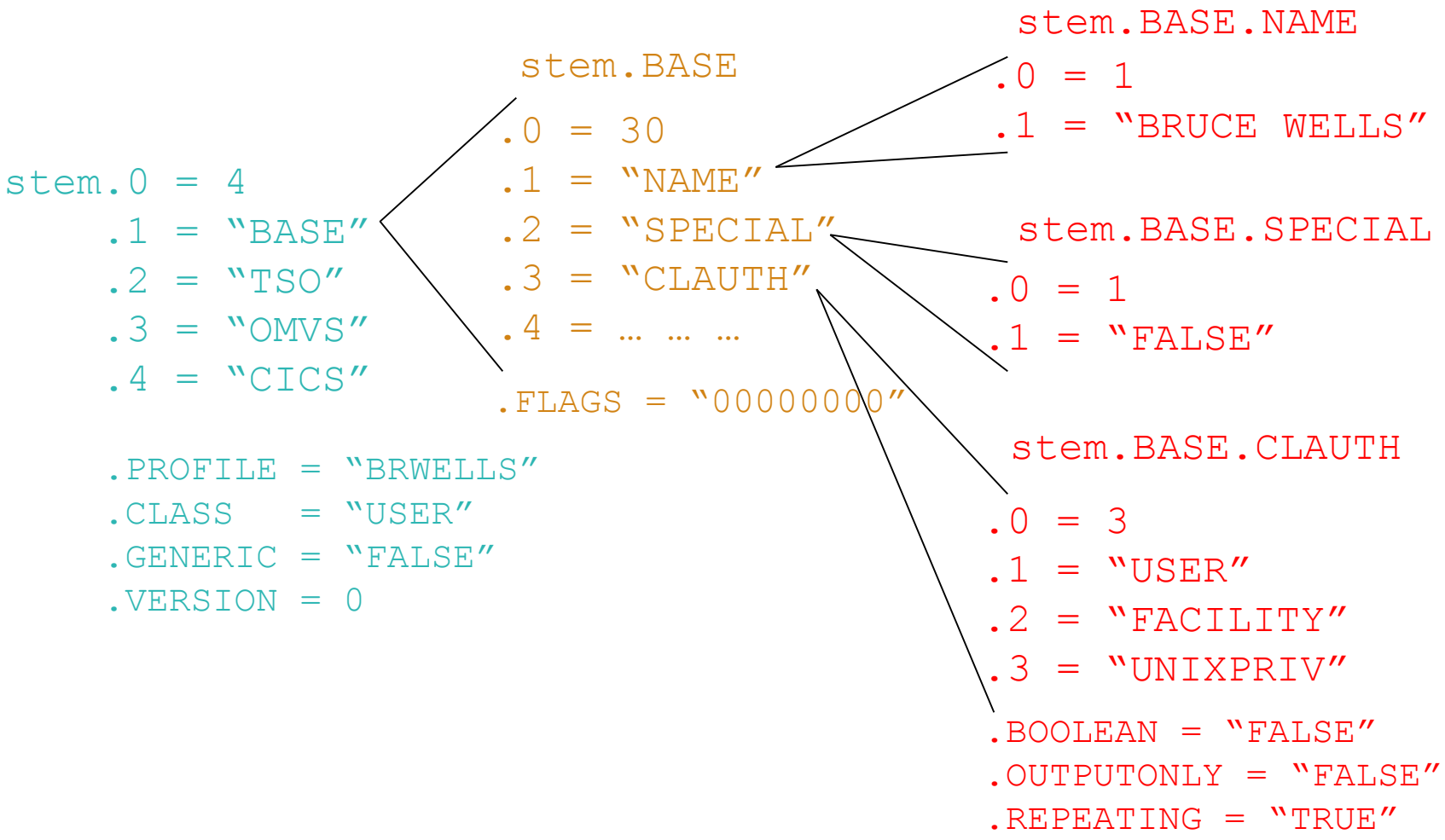
Field name	Flag byte values	ADDGROUP/ALTGROUP keyword reference, or LISTGROUP heading (for output-only fields)	Allowed on add requests	Allowed on alter requests	Returned on extract requests
SUPGROUP	'Y'	SUPGROUP(xx)	Yes	Yes	Yes
OWNER	'Y'	OWNER(xx)	Yes	Yes	Yes
TERMUACC (boolean)	'Y'	TERMUACC	Yes	Yes	Yes
	'N'	NOTERMUACC	Yes	Yes	
DATA	'Y'	DATA(xx)	Yes	Yes	Yes
	'N'	NODATA	No	Yes	

Extract?

Segment

Field

Field Name Structure



IRRXUTIL return codes

- myrc=**IRRXUTIL(*function,type,profile,stem,prefix,generic*)**
- MYRC is the return code from IRRXUTIL. It is a list of 5 numbers. If the first=0, IRRXUTIL was successful and data has been returned.

Description	RC1	RC2	RC3	RC4	RC5
Success	0	0	0	0	0
Warning, stem contained '.'	2	0	0	0	0
Bad number of parameters specified	4	Number of parms specified	Min number allowed	Max number allowed	0
Parameter Error	8	Index of bad parameter	1=Bad length 2=Bad value 3=Imcompatible with other parms	0	0
R_admin failure	12	12	R_admin safrc	R_admin racfrc	R_admin racfrsn
Environmental error	16	0=Rexx Error 4=R_admin error	For IBM support	For IBM support	0

Common Return Codes

- **0 0 0 0 0 = Success**
- **8 x y 0 0 = Error in IRRXUTIL invocation**
  - "x" – Number of the incorrect parameter
  - "y" – What's wrong
    - 1: Bad length
    - 2: Bad value
    - 3: Inconsistent with other parameters
- **12 12 4 4 4 = Profile not found**
- **12 12 8 8 24 = Not authorized to R_admin extract**

Extract NEXT for General Resource Profiles

- When extracting General Resources with EXTRACTN, start out with non generic profiles, by specifying 'FALSE' for the GENERIC parameter.
- Every time IRRXUTIL(EXTRACTN…) is called, pass in the returned 'generic' indicator (stem.GENERIC), along with the returned profile name.
- IRRXUTIL(EXTRACTN..) will automatically switch over to GENERIC profiles when it has gone through all discrete profiles.

Gotcha's…

- **Do not beat on the RACF database. For example, do not EXTRACT-NEXT all users in an attempt to find all users which belong to a given Universal Group.**
- **IRRXUTIL sets the entire stem to "" (null) before setting new data. Fields which do not exist in the extracted profile remain null.**
  - Including fields which when not null have numeric data
- **Universal Groups.**
  - Remember that a universal group profile does not contain a list of the users who are connected to the group with USE authority.
- **Discrete profiles which contain generic characters will cause the underlying R_admin service to fail if they are encountered during an EXTRACTN call.**
  - IRRXUTIL fails also
  - The only solution is to RDELETE these erroneous profiles.
  - There are few cases where discrete profiles are expected to contain generic characters and R_admin handles these properly.

IRRXUTIL Samples, from the RACF Downloads Page.

- XDUPACL.txt - A program which looks for user ACL entries which may be redundant with existing group ACL entries
- XLGRES.txt - A program which resumes the group connection of every member of a group
- XLISTGRP.txt - A program which displays a group's connected users in alphabetic order, with each user's name and connect authority
- XLISTUSR.txt - A program which displays a user's connect groups in alphabetic order
- XRACSEQ.txt - A program which re-implements the RACSEQ download to demonstrate features of IRRXUTIL
- XRLIST.txt - A program which displays the standard access list of a general resource profile with the users listed first, in alphabetic order, with the user's name, followed by the groups, in alphabetic order
- XSETRPWD.txt - A program which displays only the password-related SETROPTS options, and indicates whether password and password phrase enveloping is active
- XWHOCAN.txt - A program which displays certain users who can modify the specified profile

Using System REXX and IRRXUTIL to Write your Own RACF Health Check

Our Sample IRRXUTIL-based RACF System REXX Health Check

- **Create a Health Check which examines the profiles in the RACF STARTED class and flags as exceptions any profile which:**
  - References a user ID which does not exist
  - References a user ID which is not a PROTECTED user
- Based upon 'SYS1.SAMPLIB(HZSSXCHK)'

```
CHECK(IBMSAMPLE,RACF_STARTED_CLASS)
START TIME: 04/25/2011 15:10:58.866140
CHECK DATE: 20061219  CHECK SEVERITY: LOW
VERBOSE MODE: YES
                        RACF Started Class Report
S Started Profile    Gen   User     Group    Prot  Trust Priv  Trace
- ----------------- ----- -------- -------- ----- ----- ----- -----
  IRRDPTAB.*         Yes   STCUSER  SYS1     Yes   Yes   No    No
  JES2.*             Yes   STCUSER  SYS1     Yes   Yes   No    No
E JK.*               Yes   NOTPROT  SYS1     No    Yes   No    No
  JKOO.*             Yes   PROTECT  SYS1     Yes   Yes   No    No
E JKOOP.*            Yes   MARKN    SYS1     No    Yes   No    No
  JKOOP2.*           Yes   PROTECT  SYS1     Yes   Yes   No    No
  RACF.*             Yes   STCUSER  SYS1     Yes   Yes   No    No
  RSF1.*             Yes   STCUSER  SYS1     Yes   Yes   No    No
  RSF1WKSP.*         Yes   STCUSER  SYS1     Yes   Yes   No    No
U RSF2.*             Yes   NOSUCHID SYS1     ***   Yes   No    No
  RSF2WKSP.*         Yes   STCUSER  SYS1     Yes   Yes   No    No
E **                 Yes   IBMUSER  SYS1     No    Yes   No    Yes

* Low Severity Exception *

RACS260E One or more started tasks were found which had associated user
IDs which do not have the protected attribute.
. . .
```

Our Sample IRRXUTIL-based RACF System REXX Health Check

- **Step 1:** Initialize the System REXX environment so that it can use Health Checker services by calling HZSLSTRT

```
/*********************************************************************/
/* A System REXX Health Check must call HZSLSTRT. If this call is    */
/* not successful all IBM Health Checker for z/OS                    */
/* function calls will fail.                                         */
/*********************************************************************/
 HZSLSTRT_RC = HZSLSTRT()
 IF HZSLSTRT_RC <> 0 THEN
   DO
         SAY "HZSLSTRT RC"  HZSLSTRT_RC
         SAY "HZSLSTRT RSN" HZSLSTRT_RSN
         SAY "HZSLSTRT SYSTEMDIAG" HZSLSTRT_SYSTEMDIAG
     EXIT                      /* Exit, check cannot be performed     */
   END
```

Our Sample IRRXUTIL-based RACF System REXX Health Check

- **Step 2: Every Health Check an ENTRY code, defined when the check is registered with the Health Checker address space. We'll see later that it was one (1).**
- **Step 3: Each check is called at check initial run, each time it is scheduled to be run, and for termination. For all of the "run" cases, we'll call a routine called PROCESS_RACF_STARTED_CLASS_CHECK.**

```
/*********************************************************************/
/* Check the entry code to determine which check to process          */
/*********************************************************************/
    select
      when HZS_PQE_ENTRY_CODE = 1 then     /* RACF_STARTED_CLASS     */
         select
            when HZS_PQE_FUNCTION_CODE = "INITRUN" then
               Call Process_RACF_STARTED_CLASS_CHECK
            when HZS_PQE_FUNCTION_CODE = "RUN"     then
               Call Process_RACF_STARTED_CLASS_CHECK
            when HZS_PQE_FUNCTION_CODE = "DEACTIVATE" then nop
            otherwise say "Unexpected HZS_PQE_FUNCTION_CODE:" ,
               HZS_PQE_FUNCTION_CODE
         end                               /* END RACF_STARTED_CLASS */
```

Our Sample IRRXUTIL-based RACF System REXX Health Check

- **Step 4: Issue the check output header lines**
  - Prior to z/OS V1R12, messages had to be stored in a separate module located in an APF-authorized data set accessible
  - Starting with z/OS V1R12, you don't have to have a separate message module… messages can be issued directly!

```
call putBlankLine
call putCenteredLine 'RACF Started Class'
call putBlankLine
call putLine ,
'S Started Profile   Gen   User     Group    Prot  Trust Priv  Trace'
call putLine ,
'- ----------------- ----- -------- -------- ----- ----- ----  -----'
```

Our Sample IRRXUTIL-based RACF System REXX Health Check

- **Step 4b: The putBlankLine routine**

```
putBlankLine: procedure expose HZS_Handle


HZSLFMSG_REQUEST = 'DIRECTMSG'          /* A direct message request  */
HZSLFMSG_REASON='CHECKREPORT'           /* Report message            */
HZSLFMSG_DIRECTMSG_TEXT= '&rbl;'        /* Blank Line                */

HZSLFMSG_RC = HZSLFMSG()
IF HZS_PQE_DEBUG = 1 THEN
   DO
      SAY "HZSLFMSG RC"  HZSLFMSG_RC
      SAY "HZSLFMSG RSN" HZSLFMSG_RSN
      SAY "SYSTEMDIAG"   HZSLFMSG_SYSTEMDIAG
      IF HZSLFMSG_RC = 8 THEN
         DO
           SAY "USER RSN"    HZSLFMSG_UserRsn
           SAY "USER RESULT" HZSLFMSG_AbendResult
         END
   END
return
```

Our Sample IRRXUTIL-based RACF System REXX Health Check

- **Step 4c: The putCenteredLine routine**

```
putCenteredLine: procedure expose HZS_Handle
 parse arg text_line
 text_line_length=length(text_line)
 blanks_needed=36-(text_line_length/2)
 do i=1 to blanks_needed
    text_line= '&rbl;' || text_line
    end
 HZSLFMSG_REQUEST = 'DIRECTMSG'          /* A direct message request  */
 HZSLFMSG_REASON='CHECKREPORT'           /* Report message            */
 HZSLFMSG_DIRECTMSG_TEXT= text_line      /* Text                      */

 HZSLFMSG_RC = HZSLFMSG()
 IF HZS_PQE_DEBUG = 1 THEN
    DO
       SAY "HZSLFMSG RC"  HZSLFMSG_RC
       SAY "HZSLFMSG RSN" HZSLFMSG_RSN
       SAY "SYSTEMDIAG"   HZSLFMSG_SYSTEMDIAG
       IF HZSLFMSG_RC = 8 THEN
          DO
            SAY "USER RSN"    HZSLFMSG_UserRsn
            SAY "USER RESULT" HZSLFMSG_AbendResult
          END
    END
 return
```

Our Sample IRRXUTIL-based RACF System REXX Health Check

- **Step 4d: The putLine Routine**

```
putLine: procedure expose HZS_Handle
parse arg text_line
HZSLFMSG_REQUEST = 'DIRECTMSG'          /* A direct message request  */
HZSLFMSG_REASON='CHECKREPORT'           /* Report message            */
HZSLFMSG_DIRECTMSG_TEXT= text_line      /* Text                      */

HZSLFMSG_RC = HZSLFMSG()
IF HZS_PQE_DEBUG = 1 THEN
   DO
      SAY "HZSLFMSG RC"  HZSLFMSG_RC
      SAY "HZSLFMSG RSN" HZSLFMSG_RSN
      SAY "SYSTEMDIAG"   HZSLFMSG_SYSTEMDIAG
      IF HZSLFMSG_RC = 8 THEN
         DO
           SAY "USER RSN"    HZSLFMSG_UserRsn
           SAY "USER RESULT" HZSLFMSG_AbendResult
         END
   END
return
```

Our Sample IRRXUTIL-based RACF System REXX Health Check

- **Step 5: Loop through the STARTED class until you've processed all of the entries. If you detect an unexpected return and reason code, issue diagnostic messages.**

```
 class = 'STARTED'
 RACF.R_PROFILE = ' '
 RACF.R_GENERIC= 'FALSE'
 checkException = "NO"

Do Forever
 myrc= ,
 IRRXUTIL("EXTRACTN",class,RACF.R_PROFILE,"RACF","R_",RACF.R_GENERIC)

 /*-------------------------------------------------------------*/
 /*  Check for "end of profiles" IRRXUTIL return code.          */
 /*-------------------------------------------------------------*/
 if (word(myrc,1)=12 & word(myrc,2)=12 & word(myrc,3)=4 & ,
     word(myrc,4)=4  & word(myrc,5)=4) then do
     Leave
   end
 else if (Word(myrc,1) <> 0) Then Do
 /*-------------------------------------------------------------*/
 /*  Any other non-zero IRRXUTIL return code is an error        */
 /*-------------------------------------------------------------*/
      call irrxutil_error
      Say "Class=" class "Profile=" RACF.R_PROFILE
      Say "Generic=" RACF.R_GENERIC
      Say "Started myrc=" myrc
      Leave
   end
```

Our Sample IRRXUTIL-based RACF System REXX Health Check

- **Step 6: We've successfully extracted a STARTED profile, which contains the user ID which is associated with the started task. Save the group name, trusted flag privileged flag, trace flag and the profile name. Then call IRRXUTIL to extract the information on the user ID.**

```
RACF.U_PROFILE = RACF.R_STDATA.R_USER.1

startedTrustedFlag    =  RACF.R_STDATA.R_TRUSTED.1
startedPrivilegedFlag =  RACF.R_STDATA.R_PRIVLEGE.1
startedTraceFlag      =  RACF.R_STDATA.R_TRACE.1
startedGroupName      =  RACF.R_STDATA.R_GROUP.1

startedGenericFlag =  RACF.R_GENERIC
startedprofileName =  RACF.R_PROFILE
exceptionFlag=" "


 myrc=IRRXUTIL("EXTRACT","USER",RACF.U_PROFILE,"RACF","U_")

RACF.R_GENERIC = startedGenericFlag
RACF.R_PROFILE = startedProfileName
```

Our Sample IRRXUTIL-based RACF System REXX Health Check

- **Step 7: Flag the user ID as an exception if:**
  - **If the user ID associated with the STARTED profile does not exist**
  - **An unexpected error occurred (write diagnostic information also)**
  - **If the user ID is not PROTECTED**

```
/*-------------------------------------------------------------------*/
/*  Check for "no profile found" IRRXUTIL return code. This means */
/*  that the started profile is referencing a user ID which does  */
/*  not exist. This is an exception.                               */
/*-------------------------------------------------------------------*/
if (word(myrc,1)=12 & word(myrc,2)=12 & word(myrc,3)=4 & ,
    word(myrc,4)=4  & word(myrc,5)=4) then do
     exceptionFlag="U"
     end
else if (Word(myrc,1) <> 0) Then Do
/*-------------------------------------------------------------------*/
/*  Any other non-zero IRRXUTIL return code is an error            */
/*-------------------------------------------------------------------*/
     call irrxutil_error
     exceptionFlag="U"
     End

  if RACF.U_BASE.U_PROTECTD.1=FALSE then do
     exceptionFlag="E"
     checkException= "YES"
     end
```

Our Sample IRRXUTIL-based RACF System REXX Health Check

- **Step 8: Issue the report line message with all of the information that we have collected**

```
 data_line= ,
    LEFT(exceptionFlag,1,' ') ,
    LEFT(RACF.R_PROFILE,17,' ') ,
    LEFT(YesNo(RACF.R_GENERIC),5,' ') ,
    LEFT(RACF.U_PROFILE,8,' '),
    LEFT(startedGroupName,8,' '),
    LEFT(YesNo(RACF.U_BASE.U_PROTECTD.1),5,' '),
    LEFT(YesNo(startedTrustedFlag),5,' '),
    LEFT(YesNo(startedPrivilegedFlag),5,' '),
    LEFT(YesNo(startedTraceFlag),5,' ')

call putLine data_line
```

Our Sample IRRXUTIL-based RACF System REXX Health Check

- **Step 9: Issue the final message, which will be either “Exception Found” or “No Exception Found” and stop the check**

```
if checkException = "NO"  then do
    call putStartedClassNoException
  END
 else do
    call putStartedClassException
  END
  HZSLSTOP_RC = HZSLSTOP()             /* report check completion    */
  IF HZS_PQE_DEBUG = 1 THEN
   DO                                /* Report debug detail in REXXOUT */
     SAY "HZSLSTOP RC"  HZSLSTOP_RC
     SAY "HZSLSTOP RSN" HZSLSTOP_RSN
     SAY "HZSLSTOP SYSTEMDIAG" HZSLSTOP_SYSTEMDIAG
   END
Return
```

Our Sample IRRXUTIL-based RACF System REXX Health Check

- **Step 9b: Exception found**

```
putStartedClassException:Procedure expose HZS_Handle
HZSLFMSG_REQUEST = 'DIRECTMSG'          /* A direct message request  */
HZSLFMSG_REASON='CHECKEXCEPTION'        /* Exception                 */
HZSLFMSG_DIRECTMSG_ID='ZZZSAM01E'       /* Message identifier        */

HZSLFMSG_DIRECTMSG_TEXT=,               /* Message text              */
'One or more of the user IDs assigned in the STARTED ' ,
'class do not have the PROTECTED  attribute.'

HZSLFMSG_DIRECTMSG.EXPL=,               /* Message explanation       */
'IBM recommends assigning user IDs with the protected attribute to ',
'started tasks.'

HZSLFMSG_DIRECTMSG.SYSACT=,             /* System Action             */
'Processing continues.'

HZSLFMSG_DIRECTMSG.ORESP='None.'        /* Operator Response         */

HZSLFMSG_DIRECTMSG.SPRESP=,             /* System Programmer Response*/
'Report this issue to the security administrator.'

HZSLFMSG_DIRECTMSG.PROBD='None.'        /* Problem Determination     */

HZSLFMSG_DIRECTMSG.REFDOC='None.'       /* Reference Documentation   */
HZSLFMSG_RC = HZSLFMSG()

/* Plus the usual checking of the HZSLFMSG_RC...                     */
```

Our Sample IRRXUTIL-based RACF System REXX Health Check

- **Step 9c: No exception found**

```
 putStartedClassNoException:Procedure expose HZS_Handle
 HZSLFMSG_REQUEST = 'DIRECTMSG'          /* A direct message request  */
 HZSLFMSG_REASON='CHECKINFO'             /* Not an exception          */
 HZSLFMSG_DIRECTMSG_ID='ZZZSAM02I'       /* Message identifier        */

 HZSLFMSG_DIRECTMSG_TEXT=,               /* Message text              */
 'All of the user IDs assigned in the STARTED class have',
 'the PROTECTED attribute.'

 HZSLFMSG_RC = HZSLFMSG()

/* Plus the usual checking of the HZSLFMSG_RC...                      */
```

Registering your Check

- **The check is “registered” or defined to the z/OS Health Checker by placing an policy statement in the Health Checker PARMLIB:**

```
ADDREP CHECK(IBMSAMPLE,RACF_STARTED_CLASS)
    EXEC(HCSTART)
    REXXHLQ(MARKN)
    REXXTSO(NO)
    REXXIN(NO)
    MSGTBL(HCSTCMSG)
    ENTRYCODE(1)
    USS(NO)
    VERBOSE(YES)
    SEVERITY(LOW)
    INTERVAL(ONETIME)
    DATE(20161009)
    REASON('RACF Started Class Sample Check')
```

- **Activating the partly entry (HZSPRM$N) is activated starts the check execution:**

```
F HCMARKN,ADD,PARMLIB=$N
```

References: IRRXUTIL

- **RACF Callable Services – R_admin documentation**
  - http://publibz.boulder.ibm.com/cgi-bin/bookmgr_OS390/Shelves/ICHZBKA0
- **Command Language Reference**
  - http://publibz.boulder.ibm.com/cgi-bin/bookmgr_OS390/Shelves/ICHZBKA0
- **Macros and Interfaces – IRRXUTIL, including an exhaustive list of all REXX variables set by IRRXUTIL**
  - http://publibz.boulder.ibm.com/cgi-bin/bookmgr_OS390/Shelves/ICHZBKA0
- **RACF Downloads page – Sample R_admin extract program (RACSEQ)**
  - http://www.ibm.com/servers/eserver/zseries/zos/racf/downloads/racseq.html
- **RACF Downloads page – IRRXUTIL examples.**
  - http://www-03.ibm.com/servers/eserver/zseries/zos/racf/downloads/irrxutil.html

References: IBM Health Checker for z/OS

- **IBM Health Checker for z/OS User's Guide (SA22-7994)**
  - http://www.ibm.com/support/docview.wss?uid=pub1sa22799407
- **Exploiting the IBM Heath Checker for z/OS (Redbook)**
  - http://www.redbooks.ibm.com/abstracts/redp4590.html?Open
- **IBM Education Assistant**
  - http://www.ibm.com/software/info/education/assistant/
- **The IBM Health Checker for z/OS web site**
  - http://www.ibm.com/systems/z/os/zos/hchecker/
- **A list of all of the IBM-supplied checks** can be found at:
  - http://www.ibm.com/systems/z/os/zos/hchecker/check_table.html
- ***"An apple a day…. keeps the PMRs away! An overview of the IBM Health Checker for z/OS"***
  - z/OS Hot Topics, Issue 13, August 2005, available at http://www.ibm.com/servers/eserver/zseries/zos/bkserv/hot_topics.html
- "***RACF and the IBM Health Checker for z/OS"***
  - ibid
- ***"Personalize your RACF Checking with the IBM Health Checker for z/OS"***
  - z/OS Hot Topics, Issue 19, August 2008, available at http://www.ibm.com/servers/eserver/zseries/zos/bkserv/hot_topics.html

The IBM® Health Checker for z/OS®, IRRXUTIL, and System REXX: A Triumphant Trio!

Session 19646

Mark Nelson, CISSP®, CSSLP®
RACF Design and Development
IBM Poughkeepsie
markan@us.ibm.com