RSH

CONSULTING

RACF
Performance Tuning

February 2024

RSH Consulting - Robert S. Hansel

RSH Consulting, Inc. is an IT security professional services firm established in 1992 and dedicated to helping clients strengthen their IBM z/OS mainframe access controls by fully exploiting all the capabilities and latest innovations in RACF. RSH's services include RACF security reviews and audits, initial implementation of new controls, enhancement and remediation of existing controls, and training.

- www.rshconsulting.com
- 617-969-9050
- www.linkedin.com/company/rsh-consulting-inc.

Robert S. Hansel is Lead RACF Specialist and founder of RSH Consulting, Inc. He began working with RACF in 1986 and has been a RACF administrator, manager, auditor, instructor, developer, and consultant. Mr. Hansel is especially skilled at redesigning and refining large-scale implementations of RACF using role-based access control concepts. He is a leading expert in securing z/OS Unix using RACF. Mr. Hansel has created elaborate automated tools to assist clients with RACF administration, database merging, identity management, and quality assurance.

- 617-969-8211
- R.Hansel@rshconsulting.com
- www.linkedin.com/in/roberthansel

RACF and z/OS are Trademarks of the International Business Machines Corporation

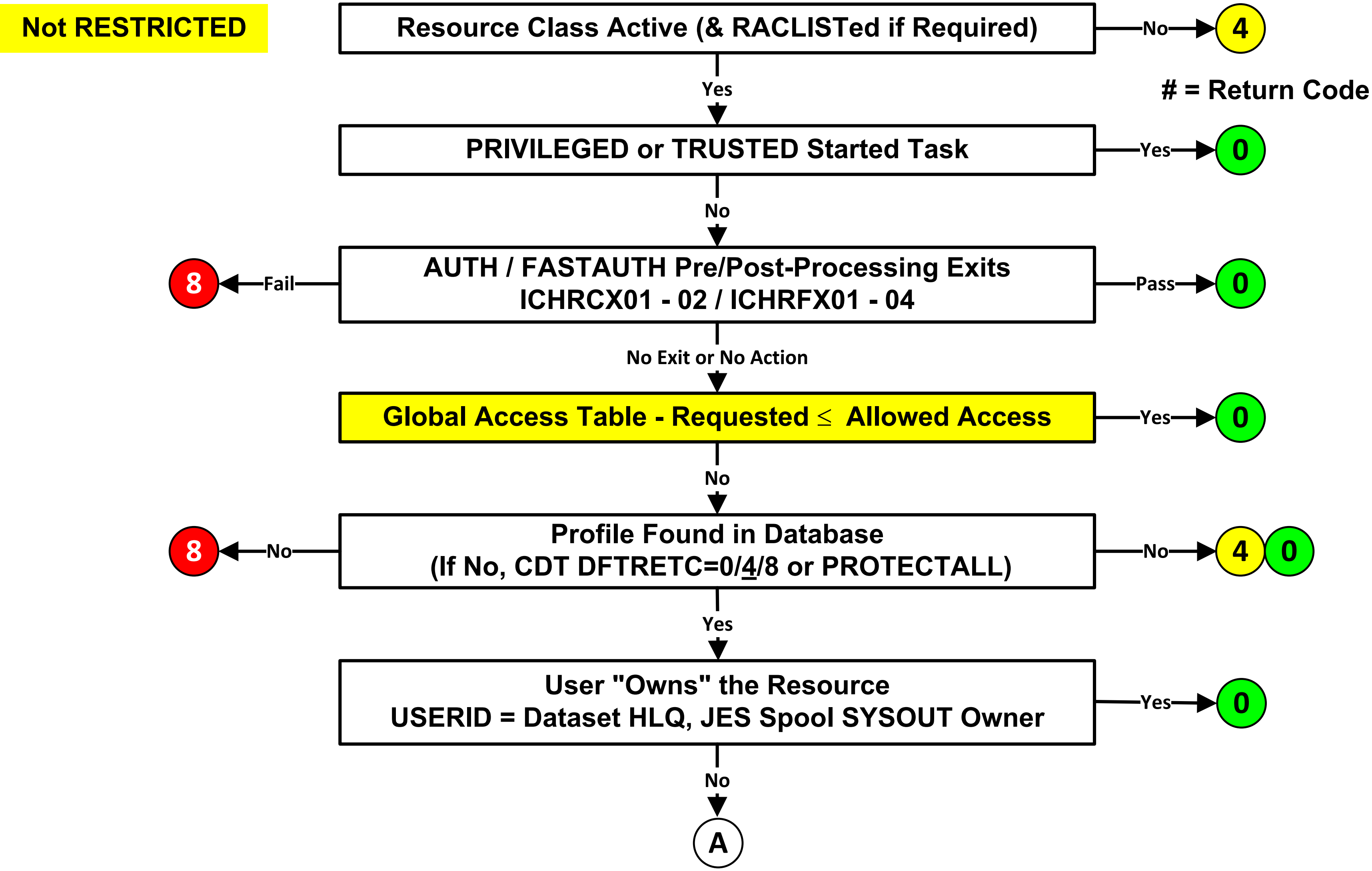
Performance Objectives

- Optimize Access Authorizations
- Expedite the Logon Process
- Minimize I/O Operations

Performance Tuning Toolkit

- Global Access Table (GAT)
- Resident Data Blocks (RDBs)
- Generic Anchor Table Entries (GATEs)
- GENLIST and RACLIST
- RACGLIST
- Global Resource Serialization (GRS)
- Sysplex Data Sharing
- Database Reorganization
- Application Identity Mapping (AIM)
- Virtual Lookaside Facility (VLF)
- Enqueue Residency

Access Authorization Decision Logic



Not RESTRICTED

Resource Class Active (& RACLISTed if Required)
No → 4
Yes ↓

# = Return Code

PRIVILEGED or TRUSTED Started Task
Yes → 0
No ↓

AUTH / FASTAUTH Pre/Post-Processing Exits
ICHRCX01 - 02 / ICHRFX01 - 04
Fail → 8
Pass → 0
No Exit or No Action ↓

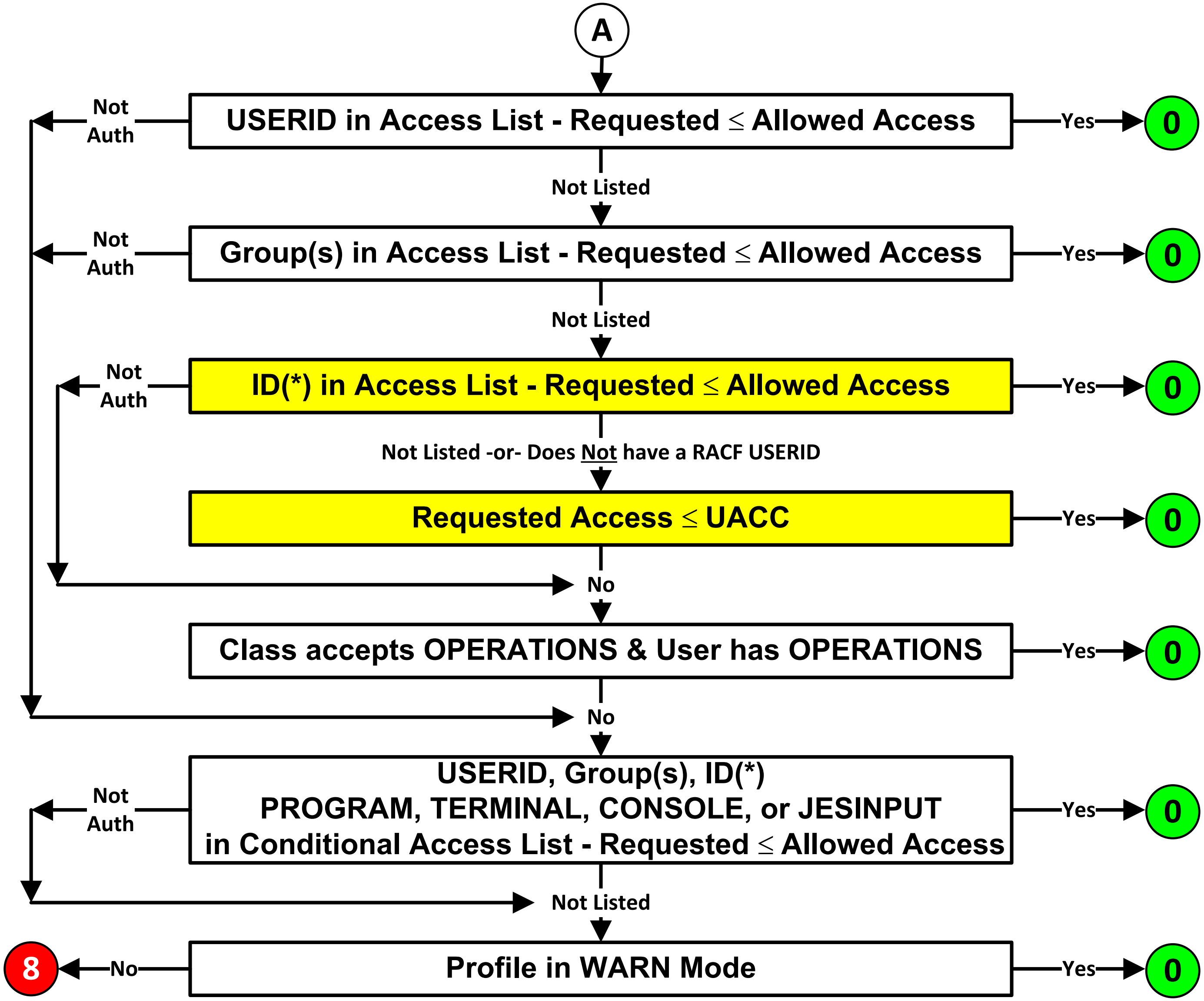
Global Access Table - Requested ≤ Allowed Access
Yes → 0
No ↓

Profile Found in Database
(If No, CDT DFTRETC=0/4/8 or PROTECTALL)
No → 8
No → 4 0
Yes ↓

User "Owns" the Resource
USERID = Dataset HLQ, JES Spool SYSOUT Owner
Yes → 0
No ↓

A

Access Authorization Decision Logic



USERID in Access List - Requested ≤ Allowed Access

Group(s) in Access List - Requested ≤ Allowed Access

ID(*) in Access List - Requested ≤ Allowed Access

Not Listed -or- Does Not have a RACF USERID

Requested Access ≤ UACC

Class accepts OPERATIONS & User has OPERATIONS

USERID, Group(s), ID(*)
PROGRAM, TERMINAL, CONSOLE, or JESINPUT
in Conditional Access List - Requested ≤ Allowed Access

Profile in WARN Mode

RACF Authorization Decision Logic

- Deactivate unused classes (be mindful of shared POSITs when deactivating)
  - Resource classes, including SECDATA and SECLABEL classes
  - Global Access Table classes

- Make access list processing efficient
  - Minimize the number of entries in access lists
    - Grant end-user access via groups instead of USERIDs
    - Remove obsolete residual entries - run IRRRID00
    - Remove redundant entries (e.g., access allowed equals UACC)
      - Exception - permission is intended to limit OPERATIONS authority
  - Minimize the number of group connects per user
  - Permit access directly to Started Task and Batch IDs to expedite their access authorization

- Reduce reliance on OPERATIONS authority
  - Implement Storage Administration authorities
  - Permit access to replace use of OPERATIONS

- Write efficient exit code

- Implement the Global Access Table (GAT)

Global Access Table

- Performance enhancement tool
  - Grants immediate access to resources without checking profiles or logging access
  - Intended to grant all users access to non-sensitive, commonly and frequently used resources
    - Includes undefined users but excludes RESTRICTED users
- Comprised of GLOBAL class profiles which contain access granting entries
  - GLOBAL class profiles are the names of other classes
    - RDEF GLOBAL DATASET
  - Entries are defined as GLOBAL profile members
    - Use ADDMEM to add entries and DELMEM to delete entries
    - Entry format: *resource-or-mask/access-level* [Ex: ADDMEM('CATLG.*'/READ)]
    - Entries can be Discrete or Generic - follows normal generic profile rules; however, for datasets, generic characters can be used in the first qualifier
    - Need not match profile(s) protecting the resource(s)
    - For datasets, appends user's USERID as the first qualifier if entry is not enclosed in quotes
    - Access-levels - ALTER | CONTROL | UPDATE | READ | NONE (not EXECUTE)
- Special Variables - Used in resource names
  - &RACUID Substitute with requesting user's USERID
  - &RACGPID Substitute with requesting user’s current connect group

Global Access Table - Sample Entries

DATASET	&RACUID.*.**	ALTER	
DATASET	&RACGPID.*.**	UPDATE	(avoid - unintended access)
DATASET	CATALOG.MASTER	READ	
DATASET	CATALOG.USER	UPDATE	
DATASET	ISPF.LIBRARY	READ	
DATASET	SDSF.LIBRARY	READ	
DATASET	SYS1.BRODCAST	READ	(UPDATE no longer required)
DATASET	SYS1.HELP	READ	
DATASET	SYS1.MACLIB	READ	
DATASET	SYS1.RACF	NONE	(precludes GAT access)
DATASET	SYS%.**	READ	(avoid - too broad)
DATASET	*.PUBLIC.**	READ	(optionally allow TSO users to share data)
DATASET	*.**.#SMSTEST	ALTER	(optional catalog/SMS testing)
FACILITY	ERBDSB.*	READ	
FACILITY	IEC.TAPERING	READ	(probably obsolete)
FACILITY	STGADMIN.ARC.ENDUSER.**	READ	
JESJOBS	SUBMIT.*.&RACUID*.&RACUID	READ	
JESJOBS	CANCEL.*.&RACUID.*	ALTER	(not needed - post RTOKEN check)
JESSPOOL	*.&RACUID.**	ALTER	
JESSPOOL	*.*.$JESNEWS.**	READ	
OPERCMDS	MVS.CANCEL.TSU.&RACUID	UPDATE	
OPERCMDS	MVS.DISPLAY.*	READ	
OPERCMDS	MVS.MCSOPER.&RACUID	READ	
SDSF	ISFCMD.DSP.*	READ	
TSOAUTH	JCL	READ	
TSOAUTH	RECOVER	READ	

Global Access Table

- Activated and managed via SETROPTS
  - SETROPTS GLOBAL(*class*) [REFRESH] | NOGLOBAL(*class*)
  - Must be refreshed if updated

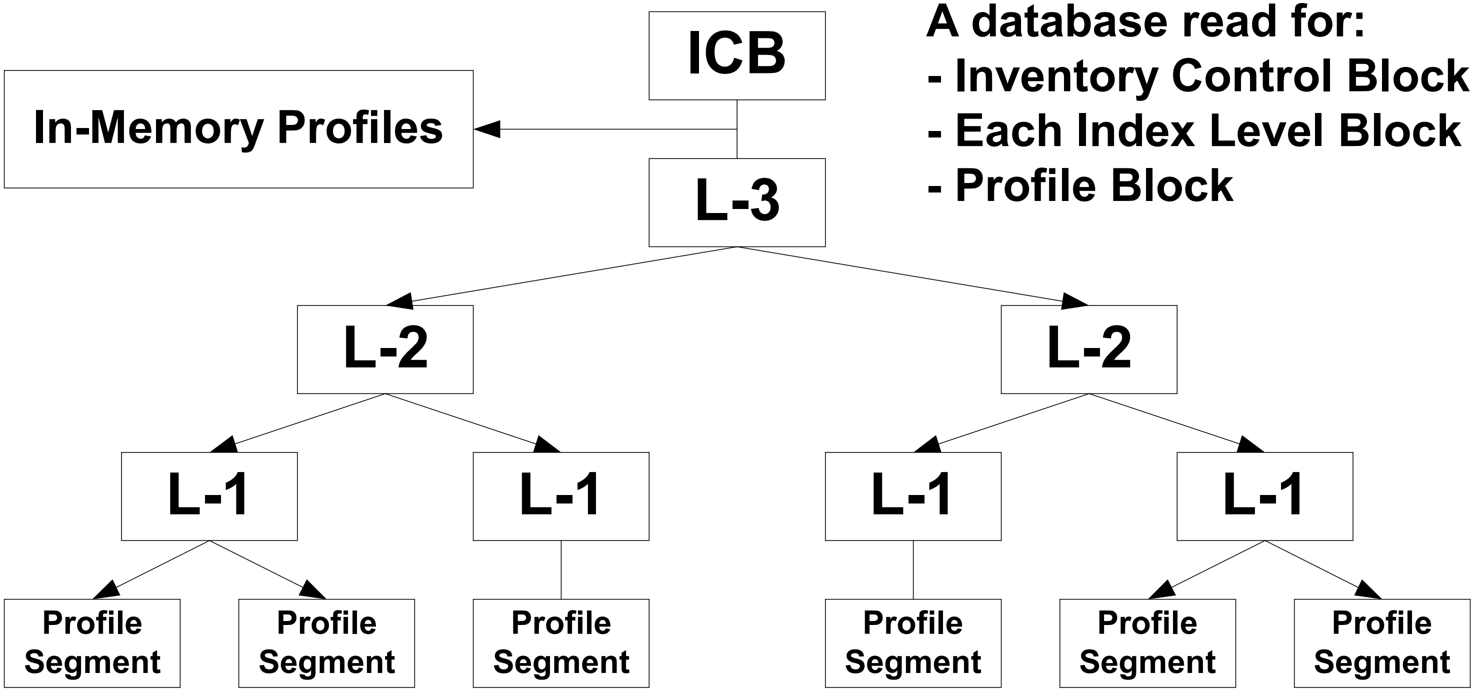
- Can be used for most resource classes except …
  - Not checked in RACROUTE REQUEST=FASTAUTH processing
  - Not checked in RACROUTE REQUEST=VERIFY processing for APPL, TERMINAL, JESINPUT, CONSOLE, APPCPORT, and SERVAUTH resources

- Keep list of entries short and efficient to minimize search

- Drawbacks
  - Precludes logging (except SETROPTS AUDIT(class) resource defines)
  - Undermines protection if allows more access than profile UACCs (common audit finding)

- If access to SYS1.BRODCAST is set to READ, RACF Administrators will need UPDATE access to it's dataset profile to maintain TSO segments

RACF Profile Retrieval



A database read for:
- Inventory Control Block (ICB)
- Each Index Level Block (L-n)
- Profile Block

- Data is written and retrieved in 4K blocks
- Individual profiles and profile segments can be greater than 4K in size and span multiple contiguous blocks, each of which requires I/O to fetch - keep profiles as small as possible

Resident Data Blocks (RDB)

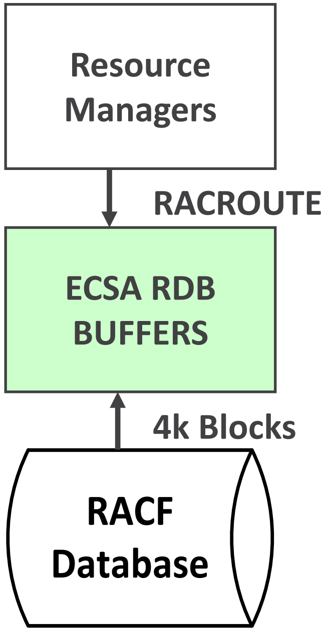
- RACF maintains buffers in Extended Common Storage Area (ECSA) to cache copies of most recently used blocks - index, Block Availability Map (BAM), and profiles

- RACF retrieves and stores a database block in an RDB before processing it

- Frequently used blocks tend to stay in these buffers (e.g., index blocks)

- Desired number of resident blocks is specified in the Database Name Table ICHRDSNT or PARMLIB(IRRPRMxx) statements

AL1(1)	Number of databases
CL44’RACF.PRIMARY’	Primary DB name
CL44’RACF.BACKUP’	Backup DB name
AL1(100)	# of Resident 4K Data Blocks
XL1’xx’	Flags



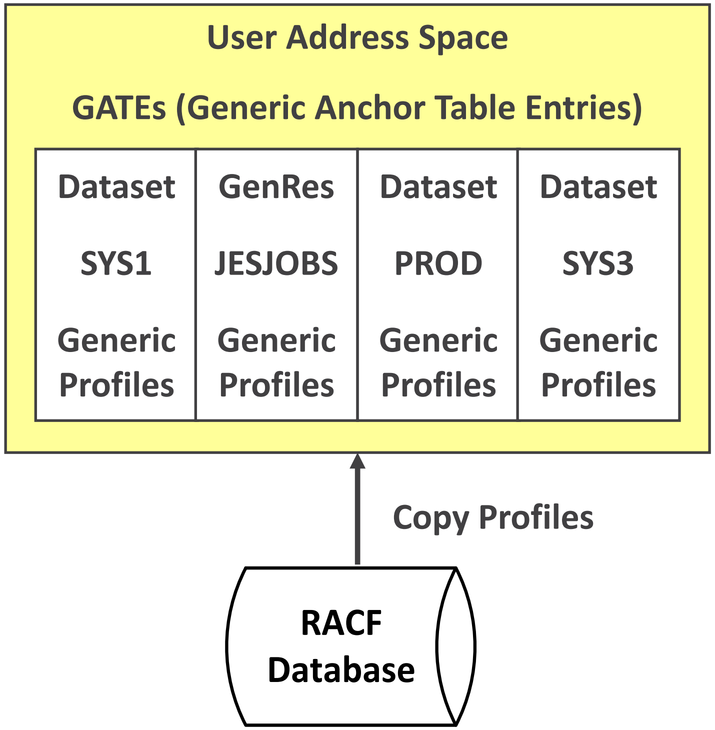
- Default/minimum number of blocks

10 / 0	Non-RACF-Sysplex (none for backup database)
50 / 50	RACF-Sysplex (+ additional 20% for backup database)

- Maximum number - 255 (recommended)

Generic Profiles Cached In Memory

- Sets of generic profiles are cached in GATEs in 64-bit memory in each user's address space
- Each GATE contains a list and copies of generic profiles for either a:
  - Dataset HLQ
  - Non-RACLIST/GENLISTed General Resource class
- Upon first access to a dataset HLQ or resource class, a list of all the associated generic profiles is retrieved and stored in a GATE along with a copy of the first generic profile needed for authorization checking
- Additional generic profiles for the same HLQ or class are retrieved as needed for authorization checking and also stored in the related GATE
- Profiles in the GATEs are used for authorization checking - not those in the RACF database



User Address Space

GATEs (Generic Anchor Table Entries)

Dataset	GenRes	Dataset	Dataset
SYS1	JESJOBS	PROD	SYS3
Generic Profiles	Generic Profiles	Generic Profiles	Generic Profiles

Generic Profiles Cached In Memory

- Once all GATEs are filled, when the next new HLQ or resource class is accessed, its profiles will replace those in the GATE containing the oldest list
  - Users randomly accessing many different HLQs and/or general resource classes could experience GATE thrashing (i.e. constant replacement)

- Dataset HLQs or general resources classes with many generic profiles take more I/O and CPU time to retrieve and load the profile list
  - CDT class profile CDTINFO(KEYQUALIFIERS(n)) limits list to first 'n' qualifiers (default = 0)

- RACF can optionally maintain up to 99 GATEs per user address space
  - Default/Minimum is 4
  - RACF operator command SET GENERICANCHOR(*options*) can change the number of GATES
    - Options: SYSTEM | JOBNAME(jobname jobname* ...) COUNT(nn)
    - Must be executed at each IPL - best to configure the RACF subsystem to execute it at start-up
  - RSH recommends setting SYSTEM COUNT to at least 20

```
SET LIST
IRRH005I (>) RACF SUBSYSTEM INFORMATION:
   ...
   GENERICANCHOR:
         SYSTEM: COUNT(04)
         JOBNAME: <NONE SPECIFIED>
```

Generic Profiles Cached In Memory

- Additions or changes to generic profiles require the copies in the GATEs to be refreshed before they become effective by one of the following methods ...
  - User can logoff and logon to refresh all GATEs
  - User can execute a LISTDSD GENERIC command to refresh the GATE for a specific HLQ

    LISTDSD DA('*hlq.anything*') GENERIC
  - SETROPTS GENERIC(*class*) REFRESH - this clears all GATEs for all users containing profiles in the designated class and requires every user to fetch and reload profiles upon next access
    - Not recommended unless there is no alternative, especially for the DATASET class

- I/O is still required for ...
  - Datasets if the RACF indicator bit is ON to look for a Discrete profile
  - General resources to check for a discrete profile before generics are checked

- Can avoid having to retrieve and load profiles into user memory by ...
  - Granting access using the Global Access Table
  - Loading profiles into memory using GENLIST and RACLIST

SETROPTS GENLIST and RACLIST

- Only apply to General Resource classes
- Intended as performance enhancement features
- Cause profiles to be stored in memory for rapid reference and to avoid I/O to the database
- Mutually exclusive SETROPTS options set for specific general resource classes
- GENLISTed and RACLISTed classes do not consume any of a user’s GATEs

SETROPTS GENLIST

- SETROPTS GENLIST(*class*)
  - GENLISTs all classes sharing the same POSIT value
- Retrieval of first Generic profile prompts retrieval and storage of a list of all Generic profiles for the class in ECSA
- Generic profiles are individually retrieved on first reference and retained in ECSA for subsequent reference
- Generic profile list and profiles in ECSA are shared by all users
- I/O still required to check for discrete profile
- Class must be defined in the CDT with GENLIST(ALLOWED)
- Refreshed with SETROPTS GENERIC(class) REFRESH
- Recommendation - use with VM related classes

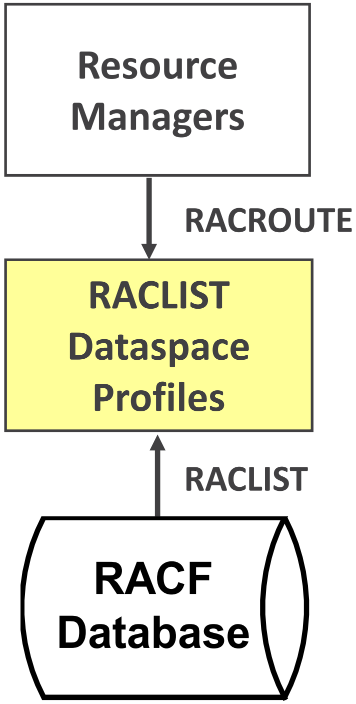
RACLIST

- All profiles for a specific class are retrieved and stored in in memory for rapid reference

- Required to exploit grouping class profiles (e.g., GCICSTRN)
  - Member class profiles and grouping class profile members are merged to form a combined list for authorization checking

- Required for RACROUTE REQUEST=FASTAUTH processing (e.g., CICS classes, UNIXPRIV, XFACILIT HealthChecker profiles)

```
SETROPTS LIST
…
SETR RACLIST CLASSES = APPL CDT DSNR FACILITY
                       STARTED TSOAUTH
GLOBAL=YES RACLIST ONLY = TCICSTRN
```



Resource Managers

RACROUTE

RACLIST Dataspace Profiles

RACLIST

RACF Database

RACLIST

- Techniques for RACLISTing a class
  - RACF command SETROPTS RACLIST(class)
    - CDT entry must specify RACLIST(ALLOWED or REQUIRED)
    - Profiles are stored in a shared dataspace
    - Class is RACLISTed on all z/OS systems sharing the RACF database
    - RACLISTs all classes sharing the same POSIT value if also defined as RACLIST(ALLOWED or REQUIRED)

```
SETR RACLIST CLASSES = APPL CDT DSNR FACILITY STARTED TSOAUTH
```

  - Resource Manager executes macro RACROUTE REQUEST=LIST,GLOBAL=YES

    CICS   DB2   IMS   VTAM   MQ
    - Profiles are stored in a shared dataspace
    - Class is RACLISTed only on the z/OS system where the Resource Manager is running
    - RACLISTs only the class specified in RACROUTE call; POSIT is ignored

```
GLOBAL=YES RACLIST ONLY = TCICSTRN
```

  - Resource Manager executes macro RACROUTE REQUEST=LIST,GLOBAL=NO

    Broadcom/CA products   Supersession
    - Profiles are stored in the resource manager's private address space and are not shared
    - RACLISTs only the class specified in RACROUTE call; POSIT is ignored

RACLIST

- RACLIST Required - CDT attribute RACLREQ=YES / RACLIST(REQUIRED)
  - If class is not RACLISTed, profiles are ignored

    APPCSERV APPCTP CRYPTOZ CSFKEYS CSFSERV DEVICES
    DIGTCERT DIGTNMAP FIELD FSACCESS FSEXEC IDIDMAP
    NODES OPERCMDS PROPCNTL PSFMPL PTKTDATA RACFHC
    RACFVARS RDATALIB SDSF SECLABEL SERVAUTH STARTED
    SYSAUTO SYSMVIEW UNIXPRIV VTAMAPPL

- SETROPTS RACLIST recommendations:

  APPL CDT CONSOLE DASDVOL DIGT Classes DSNR
  FACILITY JES classes LDAPBIND LOGSTRM MQCMDS MQCONN
  PRINTSRV RRSFDATA TSO classes TERMINAL SURROGAT

- RACLIST is not recommended for classes with profiles subject to frequent updates as this would require frequent refreshes (e.g., TAPEVOL)

- Classes RACLISTed using RACROUTE REQUEST=LIST are typically defined with RACLIST(DISALLOWED) since SETROPTS RACLIST is not necessary or desired

RACLIST REFRESH - Shared Dataspace

- Whenever profiles are created, changed, or deleted, the dataspace has to be refreshed to retrieve an updated copy of the profiles

  SETROPTS RACLIST(*class*) REFRESH

- REFRESH Considerations
  - Ensure REFRESH is performed on all systems sharing the RACF database
    - With RACF Sysplex Communications - one REFRESH does all systems
    - With RRSF Automatic Direction - one REFRESH does all RRSF nodes
  - One REFRESH does all classes with the same POSIT value (e.g., all IBM default CICS classes have POSIT 5)
  - REFRESH warning
    - For changes made to SETROPTS RACLISTed Member class profiles, RACF issues message

      ```
      ICH11009I RACLISTED PROFILES FOR class WILL NOT REFLECT THE
      UPDATE(S) UNTIL A SETROPTS REFRESH IS ISSUED.
      ```

    - No warning is given for ..
      - Changes to Grouping class profiles
      - Changes to profiles in classes RACLISTed by RACROUTE REQUEST=LIST,GLOBAL=YES
  - With RACLIST REFRESH for a Member/Grouping class pair, if the total number of access list entries in an individual merged profile exceeds 7,200, the RACLIST will abend

RACLIST REFRESH - Local (GLOBAL=NO)

- For locally RACLISTed classes, whenever profiles are created, changed, or deleted, the profiles stored in the resource manager's private address space have to be refreshed to obtain an updated copy of the profiles
  - Some resource managers provide commands to refresh profiles
    - Supersession - NAM RACLIST command
  - Resource managers that do not provide refresh commands have to be stopped and restarted to obtain updated profiles

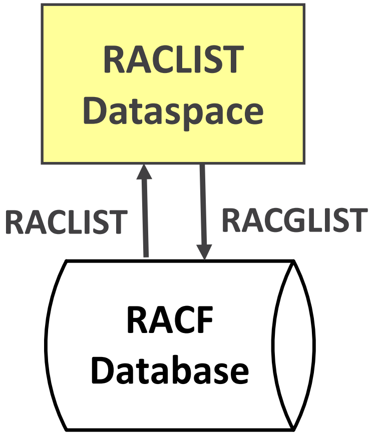
- Broadcom/CA product general resource classes
  - Classes are defined in CA Common Services (CCS) - Started Task usually named CAS9
  - Classes defined with option FASTAUTH=YES are locally RACLISTed and use RACROUTE REQUEST= FASTAUTH to process access requests with LOG=NONE
  - CA resource managers do not provide refresh commands
  - CCS's CAIRACF DD statements can change FASTAUTH settings to NO (use REQUEST=AUTH)

```
RACFCLASS PANEL,PA@EL,FASTAUTH=NO
```

  - Once changed, the class can be SETROPTS RACLISTed and refreshed as usual
    - Class CDT definition must be changed to RACLIST(ALLOWED)
  - This change also enables the use of AUDIT and GLOBALAUDIT to log access activity, enables zSecure Access Monitor to record access activity, and allows use of the GAT for granting access

RACGLIST Class

- Stores RACLISTed profiles in a post-processed form for quick re-loading (1) at IPL, (2) upon initial RACROUTE REQUEST=LIST,GLOBAL=YES, and (3) during RACLIST REFRESH
- During RACLIST REFRESH for z/OS systems sharing a database *with Sysplex communications*, the first system builds the dataspace and then stores a copy of it in the form of RACGLIST profiles for the other systems to simply retrieve and load
  - All systems will have identical dataspaces to ensure consistency in authorization checks
  - Systems sharing the RACF database must be in the same GRS complex and GRS major name SYSZRAC2 must not be in the exclusion list
- Activated for a class by defining a matching RACGLIST profile
  - SETROPTS CLASSACT(RACGLIST)
  - RDEFINE RACGLIST *member-class-name*
- REFRESH builds profiles named *class-name*_00001 - *nnnnn*
- Profiles are updated by SETROPTS RACLIST(class) REFRESH
- Ensure RACF database has sufficient space for RACGLIST profiles
- Note: IPLs no longer cause refresh of RACGLISTed classes



RACF Database Sharing

- Sharing a RACF database without Global Resource Serialization (GRS)
  - RACF uses exclusive hardware RESERVEs to serialize the database for most updates
  - System holding an exclusive RESERVE locks out other systems until it has processed all its update requests
  - Lock is on the entire DASD volume

- Global Resource Serialization (GRS)
  - Converts RESERVEs to global ENQs
  - Each system given exclusive control for one update request at a time
  - Only locks the RACF database - not the entire DASD volume
  - Avoids contention and monopolization
  - PARMLIB(GRSRNLxx) conversion entry

```
RNLDEF RNL(CON) TYPE(GENERIC) QNAME(SYSZRACF)
```

  - Restrictions
    - All z/OS systems must be part of the same GRS complex
    - Cannot be used when sharing a RACF database with a z/VM system
  - GRS is required for RACGLIST and RACF Sysplex Data Sharing

RACF Sysplex Data Sharing

- Uses Cross-system Coupling Facility (XCF - shared Sysplex-wide) as large store-through cache for Resident Data Blocks (can even improve performance for a standalone system)
  - Caches ICB, index, and profile data blocks
- Enabled by ICHRDSNT or PARMLIB(IRRPRMxx) option on first database entry

XL1’x0’	No Sysplex
XL1’x8’	RACF-Sysplex data communication without data sharing
XL1’xC’	RACF-Sysplex data communication with data sharing

- Coupling Facility Resource Manager (CFRM) sets cache policy
- To assist in calculating the coupling facility size for RACF, go to http://www.ibm.com/systems/support/z/cfsizer/racf/
- If feasible, specify size large enough to hold all index blocks plus all data blocks for non-RACLISTed resource classes
- To obtain Coupling Facility size and utilization information for the Primary RACF database, enter the following operator command (assumes a single dataset database):

```
D XCF,STR,STRNM=IRRXCF00_P001
```

RACF Database Reorganization

- Over time, administrative actions have the following effect …
  - Index entry additions fill an index block to overflowing requiring a block split leaving the two index blocks half empty, wasting both database and buffer space
  - Profile and segment deletions can empty all but small percentage of some blocks, wasting both database and buffer space
  - Newly added profile segments (e.g., TSO) get stored in different blocks than the related profile, thereby requiring more I/O to fetch, potentially slowing logons
  - Creating and deleting profiles causes fragmentation of free space making it difficult for RACF to find contiguous blocks for storing large profiles as the database nears full capacity

- IRRUT400 utility - reorganizes and, as needed, resizes the database - run periodically
  - Aligns index and associated profile blocks in sequential order
  - Fills in data blocks eliminating wasted space and fragmentation
  - Rebuilds BAM blocks, thereby eliminating any prior errors
  - Compresses the index and corrects upper level index errors
  - Optionally adds free space to index blocks for subsequent growth
  - Optionally places a profile and all its segments in the same block(s)
  - Optionally increases the size of a database that has exceeded 85% capacity or decreases the size of a database that is using less than 30% capacity (aim for 50-60% in a resize)

Logging

- Use the following logging options only when necessary for essential security oversight or temporarily for remediation
  - SETROPTS LOGOPTIONS(ALWAYS(*class*) | SUCCESSES(*class*))
  - SETROPTS OPERAUDIT
  - Resource AUDIT(SUCCESSES(READ))
  - Resource GLOBALAUDIT(SUCCESSES(READ))
  - User UAUDIT
    - Problematic if user makes extensive use of Unix File System objects or encryption services

Statistics

- Eliminate the collection of resource access statistics which have little or no value
  - SETROPTS STATISTICS(*class*) | NOSTATISTICS(*class*) Option - recommend NOSTATISTICS(*)
  - Access counts kept only on Discrete profiles
  - Not incremented for GAT permitted access or RACLISTed profiles
  - May not be accurate in a shared database environment
  - Increases CPU processing to calculate and I/O to update the profile
- Update Statistics in the backup database as needed - ICHRDSNT or PARMLIB(IRRPRMxx) option

XL1’0x’	No updates are duplicated in the backup database (default) - backup database is inactive
XL1’8x’	Updates other than statistics are duplicated (recommended)
XL1’Cx’	Updates including statistics are duplicated (avoid - increases I/O to update profiles)

- Limit updates to user logon statistics to only once per day (e.g., LAST-ACCESS)
  - Implemented via APPL class profiles for associated applications
  - Specify APPLDATA('RACF-INITSTATS(DAILY)') in APPL profile to activate
  - ICHRIX01 exit can add APPL values to all RACROUTE REQUEST=VERIFY calls without a value to better leverage this feature
- SETROPTS INACTIVE(*nn*) results in the update of logon statistics in the backup database for the first logon of the day, increasing I/O and logon lag time

Application Identity Mapping (AIM)

- Identity mapping is required when the corresponding RACF identity must be determined (e.g., Unix 'ls' command - display the RACF USERID and Group for the corresponding Unix Owner UID and Group GID)

- Options for z/OS Unix identity look-up
  - Find, fetch, and examine the OMVS segment of every user or group (not desirable)
  - UNIXMAP Class
    - Class must be activated to be used for mapping
    - Contains profiles in the form Unnn and Gnnn, where 'nnn' is a UID or GID
    - Users and groups are 'permitted' access to signify UID and GID assignment
    - Profiles are automatically maintained when OMVS segments are created or altered via RACF commands
  - Application Identity Mapping (AIM) (recommended)
    - Restructured database with mapping index structure - faster look-ups
    - Implemented using IRRIRA00 utility in stages 1 to 3 (stage 0 = no AIM structure)
    - Replaces UNIXMAP profiles, as well as profiles in classes NOTELINK (SNAME - LNOTES segment) and NDSLINK (UNAME - NDS segment)
    - Enables use of UID(nnn) and GID(nnn) with the SEARCH command
    - Required to use certain Unix control options (e.g., UNIXPRIV SHARED.IDS and FACILITY BPX.UNIQUE.USER)

- Additionally, cache UID and GID mappings in VLF

Virtual Lookaside Facility (VLF)

- VLF can cache RACF information for reuse
  - Accessor Environment Elements (ACEEs)
  - Group tree
  - z/OS Unix mappings of UIDs and GIDs to USERIDs and Groups
  - z/OS Unix User Security Packets (USPs)

- MAXVIRT parameter - VLF Maximum Virtual Storage
  - Optionally specified in PARMLIB(COFVLFxx) for each VLF CLASS
  - MAXVIRT(*nnnnnn*) - 4K block increments
    - Default: 4096
    - Range: 256 - 524288
  - Default normally sufficient
  - Monitor VLF use - SMF record type 41, subtype 3

Virtual Lookaside Facility (VLF)

- Accessor Environment Elements (ACEEs)
  - ACEE is created during logon process - contains user's attributes, lists of groups, and logon characteristics (e.g., Point-of-Entry (POE), application)
  - Caching avoids repeated retrieval of the user profile to build ACEEs for subsequent logons
  - PARMLIB(COFVLFxx) entry

```
CLASS NAME(IRRACEE)
      EMAJ(ACEE)
```

  - Most changes to a user profile cause a purge of some or all cached ACEEs for that user
    - Systems sharing a RACF database without Sysplex Communications purge all ACEEs for a user change
  - Refresh of the following classes causes a purge of all cached ACEEs

```
APPCPORT   APPL   CONSOLE   JESINPUT   MFADEF   SERVAUTH   TERMINAL
```

Virtual Lookaside Facility (VLF)

- Group tree
  - Used to determine scope-of-groups for Group-level authorities

    SPECIAL OPERATIONS AUDITOR
  - Caching avoids repeated retrieval of group profiles and tree reconstruction
  - Most important if group authority is used extensively
  - PARMLIB(COFVLFxx) entry

```
CLASS NAME(IRRGTS)
      EMAJ(GTS)
```

Virtual Lookaside Facility (VLF)

- z/OS Unix mappings of UIDs and GIDs to USERIDs and Groups
  - Mappings save the associated USERID or Group for a UID or GID (e.g., 'ls' command)
  - Caching avoids repeated retrieval of mapping information
    - Recommended even with AIM restructured database
  - PARMLIB(COFVLFxx) entry

```
CLASS NAME(IRRGMAP)
          EMAJ(GMAP)
CLASS NAME(IRRUMAP)
          EMAJ(UMAP)
```

- z/OS Unix User Security Packets (USPs)
  - USP is created when user dubs (invokes z/OS Unix function)
  - Caching avoids repeated rebuilding of USPs during subsequent dubbing
    - Especially helpful for applications using thread-level security
  - PARMLIB(COFVLFxx) entry

```
CLASS NAME(IRRSMAP)
          EMAJ(SMAP)
```

Enqueue Residency - IEAOPTxx ERV Parameter

- Contention issue - low priority TSO user or batch job gets swapped out while still holding an enqueue on SYSZRACF or a hardware RESERVE on the RACF database volume, and thereby holds up other address spaces and systems waiting on RACF

- Solution - grant more CPU Service Units to address spaces enqueued on system resources or holding hardware RESERVEs enabling them to complete work before being swapped out

- PARMLIB(IEAOPTxx) - ERV parameter
  - Range: 0 - 999999
  - Default: 500
  - Recommended: 40000 - 50000

RACF Commands and Utilities

- Avoid use of commands and utilities that are I/O or processing intensive during peak system activity periods (especially morning logon)

  LISTUSER *   LISTGRP *   RLIST class *

  LISTDSD with ID(), PREFIX(), or DSNS

  SEARCH with NOMASK and AGE, USER, or WARNING

  SETROPTS GENERIC(class) REFRESH   - especially DATASET

  SETROPTS RACLIST(class) REFRESH   - especially classes with many profiles

  Large batches of commands   - especially CONNECTs and REMOVEs

  ICHDSM00 with FUNCTION RACGRP or RACUSR

  IRRUT100

  IRRUT200 to copy the live RACF database

  IRRUT200 to analyze the live RACF database in place (use off-line IRRUT200 backup instead)

  IRRUT400 using live RACF database, especially with LOCKINPUT (use backup instead)

  IRRDBU00 using live RACF database, especially with LOCKINPUT (use backup instead)

  RACF admin product extracts using live RACF database (use backup instead)

- Specify parameter NOYOURACC (or NOY) on RLIST commands to avoid retrieval and RACLIST processing of all grouping class profiles simply to determine your access

Miscellaneous

- Avoid activating SETROPTS CATDSNS as it prompts additional checks with every dataset access authorization check to verify the dataset is cataloged or for overrides

- Keep the RACF database clean of unnecessary permissions and obsolete USERIDs, groups, and resource profiles
  - Avoids wasted space in data blocks and cache
  - Reduces processing for IRRDBU00 unload and RACF admin product extracts

- RACF database placement
  - Place each database dataset on a separate volume
  - Isolate the database datasets from other files or place them with infrequently accessed files

- Split RACF database into multiple datasets - up to 90
  - Requires implementation of RACF Range Table ICHRRNG or PARMLIB equivalent
  - Advantages - spreads workload across multiple DASD devices; each dataset gets its own I/O queue and set of RDBs
  - Disadvantages - complex; more datasets to manage, backup, etc.; requires all-way IPL to change configuration
  - Avoid implementing - only helpful with extremely large databases