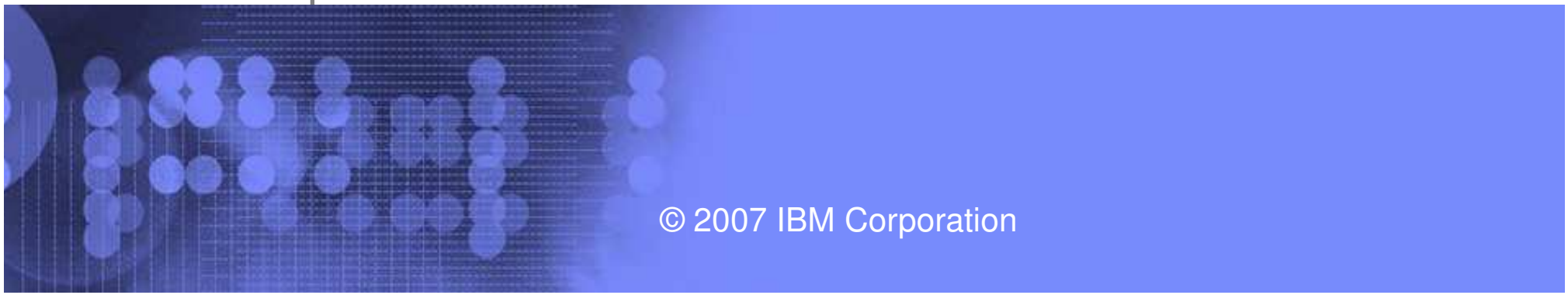
RACF®: More Dynamic Than Ever

- Dynamic Class Descriptor Table
- RACF Router Table Enhancements
- Dynamic Templates

Session RAA2

Laurie Ward
IBM® Corporation
Poughkeepsie, NY

LWard@us.ibm.com
(845) 435-8028



Trademarks

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

- CICS*
- DB2*
- IBM*
- IBM (logo)*
- OS/390*
- RACF*
- z/OS*

* 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.

Session Objectives

Things you will learn:

- RACF Class Descriptor Table
  - How it is used and updated
  - How to use the Dynamic Class Descriptor Table
- RACF Router Table
  - How it is used and updated
  - Fewer reasons to update it
- RACF Templates
  - How they are used and updated
  - How to use Dynamic Template support

V1R6

V1R6

V1R5

Dynamic Class Descriptor Table

z/OS V1R6 and later

#1 requested requirement for RACF

Overview – What is the RACF CDT?

- CDT – **C**lass **D**escriptor **T**able
  - A table of all general resource classes defined to RACF
    - Examples: TERMINAL, DASDVOL, SECLABEL
  - Attributes of each class
    - Length of resource names
    - What characters can be in resource names
    - Many other processing characteristics
  - Does not include DATASET, USER, or GROUP classes

Overview – Who uses the RACF CDT?

- RACF uses it for:
  - Authorization checking
  - Commands for defining resources
    - RDEFINE, RALTER, RDELETE, PERMIT, others
- Other z/OS® products
- Vendor products
- Local applications

Overview – Who updates the RACF CDT?

- RACF updates the IBM-supplied class list with new releases and sometimes in the service stream
  - For new RACF functions
  - For new functions in other z/OS products
- Some vendor products require you to add classes to the CDT to use their products
- Optionally, you add classes when adding new CICS® regions or new DB2® subsystems

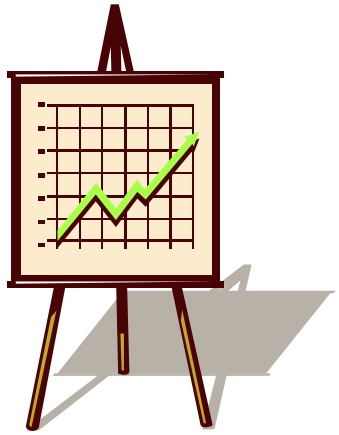
Overview of Problem and Solution

- **To update the RACF class descriptor table and router table, the installation must:**
  - Write assembler code
  - Assemble and link edit modules
  - IPL the system
    - Availability problem if running 24x7 production

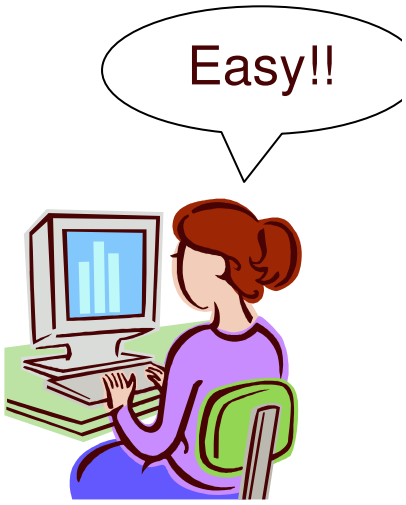
- **Solution in z/OS V1R6:**
  - Dynamic Class Descriptor Table
  - Router Table - update only for exceptions

Value of Dynamic CDT Support

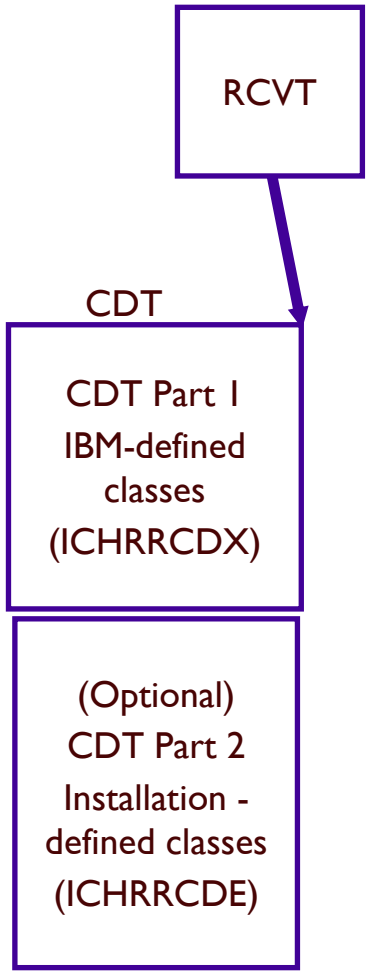
- Availability
  - No IPL necessary to add, update, or delete an installation-defined class
  - Class becomes a ‘permanent’ part of the RACF CDT



- Ease of Use
  - RACF commands can be used to add an installation-defined class
    - No assembler coding required
  - No update to RACF router table required when adding an installation-defined class
  - Easier to change attributes of a class



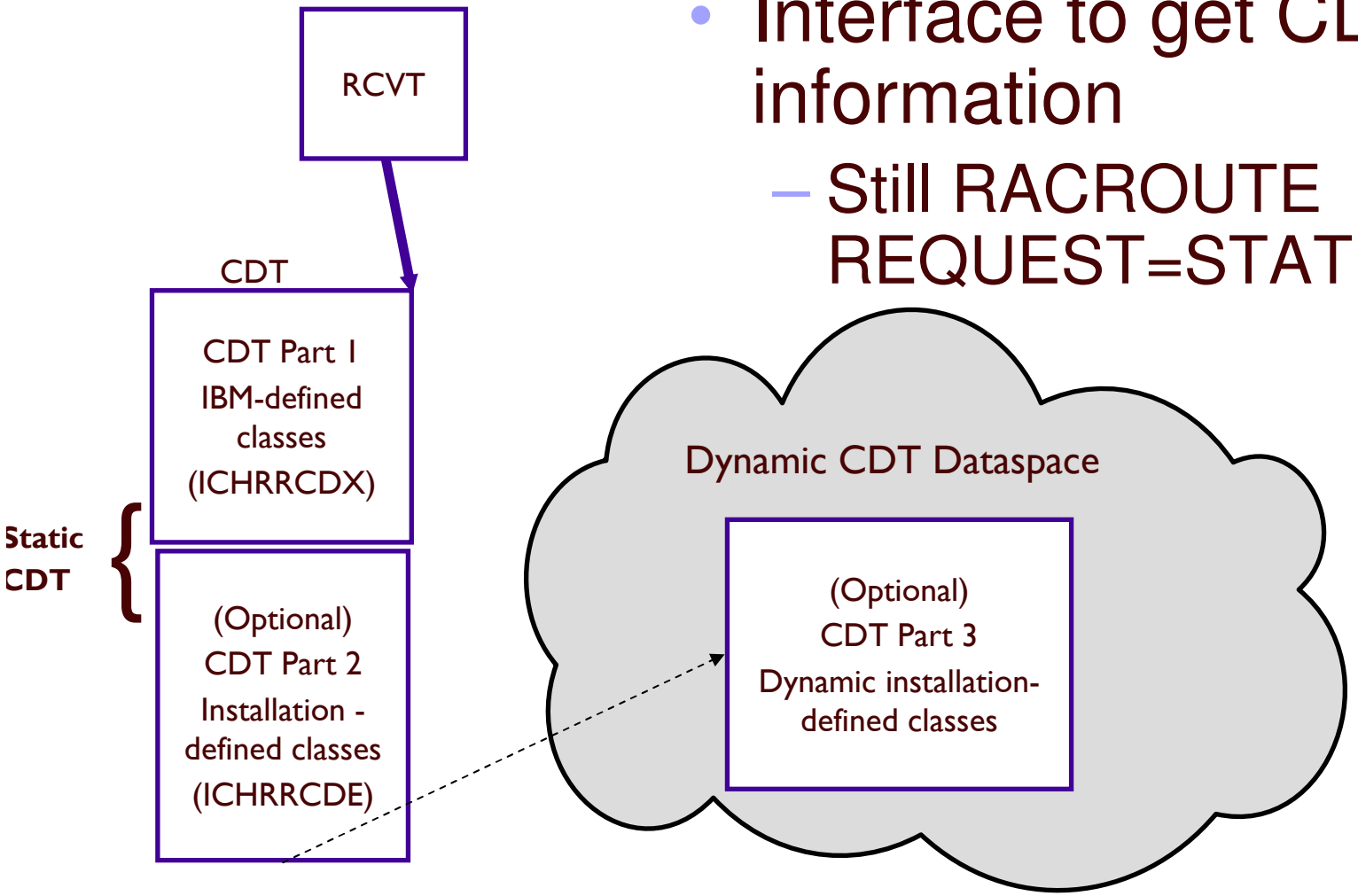
How CDT works prior to z/OS V1R6



- Interface to get CDT information
  - RACROUTE REQUEST=STAT macro
- Unintended interface
  - RCVTCDTP pointer
- Stabilized interface
  - RACSTAT macro

How Dynamic CDT works

- Interface to get CDT information
  - Still RACROUTE REQUEST=STAT macro



Dynamic CDT Usage & Invocation

- Defining a dynamic class
- Putting a dynamic class in the CDT
- Using a dynamic class
- Task shift for defining new RACF classes
- Considerations for existing installation-defined classes
- RACROUTE REQUEST=STAT enhancement

Defining a Dynamic Class

- Use IBM class named CDT to create a class definition – a profile to represent a dynamic class
- Use new segment CDTINFO to define class attributes

RDEFINE CDT *dyn-class-name* UACC(NONE) CDTINFO(*class-attribute-1 class-attribute-2 ...*)

RDEFINE CDT HORSES8 UACC(NONE) CDTINFO(DEFAULTUACC(NONE) FIRST(ALPHA) MAXLENGTH(200) OTHER(ALPHA,NUMERIC) POSIT(301) RACLIST(REQUIRED))

Defining a Dynamic Class…

- CDTINFO keyword on RDEFINE and RALTER has 19 keywords for defining class attributes

CASE	KEYQUALIFIERS	OTHER
DEFAULTRC	MACPROCESSING	POSIT
DEFAULTUACC	MAXLENGTH	PROFILESALLOWED
FIRST	MAXLENX	RACLIST
GENLIST	MEMBER	SECLABELSREQUIRED
GENERIC	OPERATIONS	SIGNAL
GROUP		

V1R8

- Command keywords correspond to keywords on ICHERCDE macro (used for static CDT)

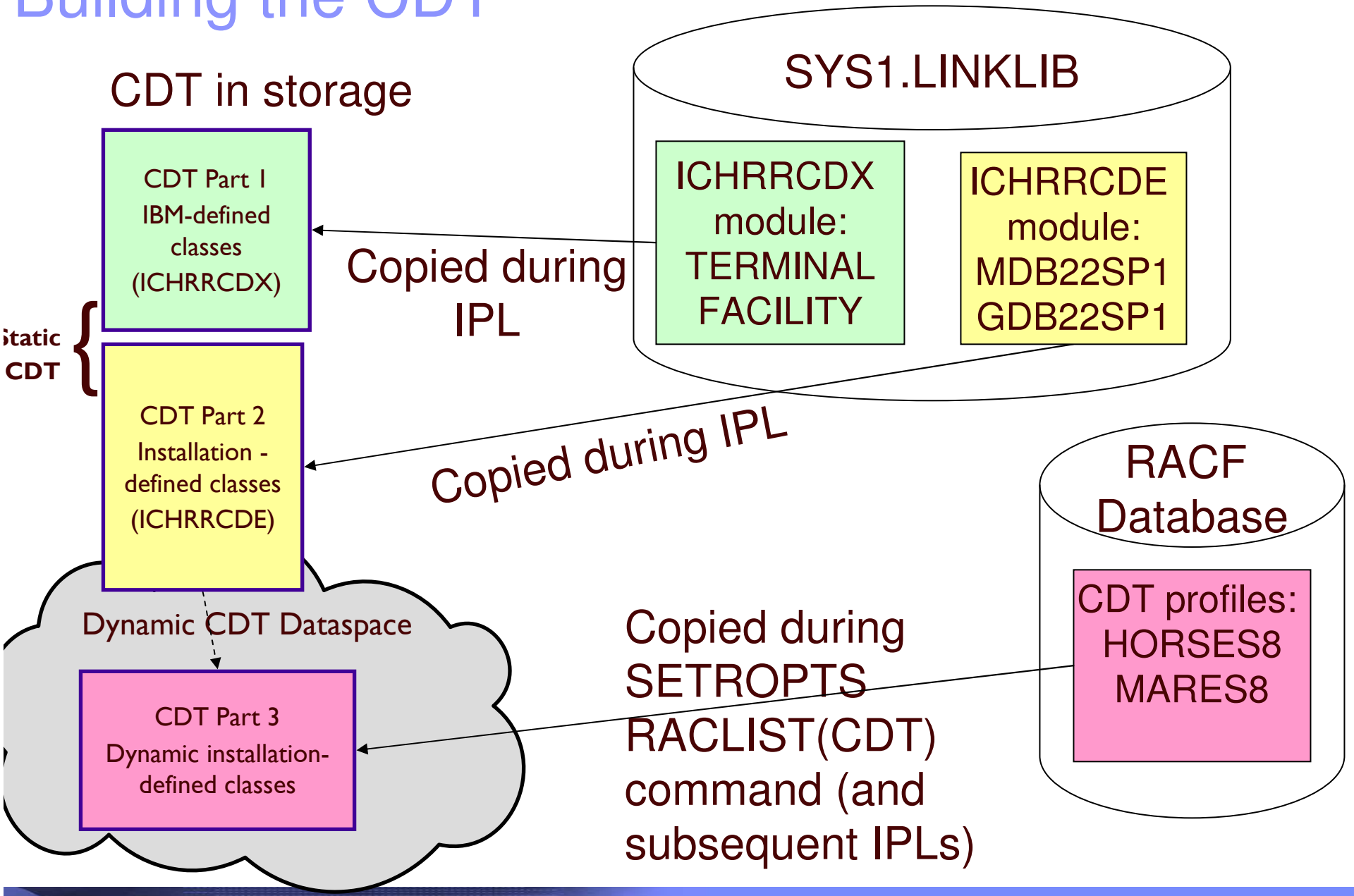
Putting a Dynamic Class into the CDT

- Use SETROPTS RACLIST command to build the Dynamic CDT

  SETROPTS CLASSACT(CDT) RACLIST(CDT)

  - This allows the HORSES8 class to be recognized as a RACF general resource class.
- Data space is created for Dynamic CDT which is a logical extension of the static CDT
- Next IPL, RACF remembers CDT class was RACLISTed
  - Dynamic CDT is built during RACF initialization

Building the CDT



Using a Dynamic Class

- Use existing RACF commands which accept a general resource class name

RDEFINE RALTER RLIST RDELETE
SEARCH SETROPTS PERMIT
ADDSD FCLASS(class-name)
ALTDSD FCLASS(class-name)
RVARY INACTIVE NOCLASSACT(class-namelist)
ADDUSER CLAUTH(class-name)
ALTUSER CLAUTH(class-name)

- Examples

RDEFINE HORSES8 HORSEY1 UACC(NONE)
SEARCH CLASS(HORSES8)
SETROPTS CLASSACT(HORSES8) RACLIST(HORSES8)

Summary of Steps to Create a Dynamic Class

1. Define class attributes – profile in CDT class

   RDEFINE CDT HORSES8 UACC(NONE) CDTINFO(
   DEFAULTUACC(NONE) FIRST(ALPHA) MAXLENGTH(200)
   OTHER(ALPHA,NUMERIC) POSIT(301) RACLIST(REQUIRED))

2. Build the Dynamic CDT

   SETROPTS CLASSACT(CDT) RACLIST(CDT)

   Or Rebuild the Dynamic CDT

   SETROPTS RACLIST(CDT) REFRESH

3. Use new class in existing RACF commands

   RDEFINE HORSES8 HORSEY1 UACC(NONE)

   SEARCH CLASS(HORSES8)

   SETROPTS CLASSACT(HORSES8) RACLIST(HORSES8)

Task Shift for Defining New RACF Classes

- Defining new RACF classes was previously a system programmer task
- Security administrators now have authority to define new RACF classes
  - Commands can be issued by users with RACF SPECIAL attribute
  - Authority can be delegated
    - CLAUTH to CDT class
    - Field level access to CDTINFO segments

Considerations for Existing Installation-defined Classes

- You can migrate your existing installation-defined classes (in ICHRRCDE) to the Dynamic CDT
  - Program is available on the RACF website to help translate definitions from ICHRRCDE module into commands to create dynamic classes
    - CDT2DYN EXEC (written in REXX)
  - For this migration, a 'duplicate' class is allowed to exist in ICHRRCDE and Dynamic CDT
    - Dynamic class definition takes precedence

What is Not Changing?

- IBM-defined classes are not dynamic
  - If IBM updates the IBM-defined classes, a new ICHRRCDX is shipped, and an IPL is still required
    - Usually other code is shipped which supports the new classes

RACROUTE REQUEST=STAT Enhancement

- NEXT= keyword allows sequential search of classes in CDT

```
RACROUTE REQUEST=STAT,
 NEXT=CLAS_NAME, COPY=CDT_INFO,
 COPYLEN=COPY_LEN, RELEASE=7709
```

RACF Router Table Enhancements

z/OS V1R6 and later

RACF Router Table

- For RACROUTE calls, the RACF router table determines if RACF is called
  - Based on the CLASS, REQSTOR, and SUBSYS keywords
- Router Table entry was required prior to z/OS V1R6
  - For each class in the CDT
  - For each combination of REQSTOR/SUBSYS
- To update the Router Table you must:
  - Code, assemble and link edit module ICHRFR01
  - IPL the system !
- Router table is still not dynamic, but……

RACF Router Table Changes in z/OS V1R6

- Default behavior changed
  - If no router table entry found, RACF is called
- Router table entry not required for:
  - Installation-defined classes
  - REQSTOR/SUBSYS combinations
  - Dynamic classes
  - IBM classes (ICHRFR0X no longer shipped)
- Installation-defined Router table still allowed
  - Only required if you need to specify ACTION=NONE so that RACF calls are bypassed for a class or a SUBSYS/REQSTOR combination
  - Still requires assembly and link edit of ICHRFR01
  - Still requires IPL

Dynamic Templates

z/OS V1R5 and later

RACF Templates Overview

- Map how profiles are written on the RACF database
- Updated to add new segments or fields to the RACF database
- Exist in three places:
  - The latest version shipped with RACF (IRRTEMP1)
  - The version on the database, written there by utility IRRMIN00
    - PARM=NEW initialize new database
    - PARM=UPDATE update templates on existing database
  - The in-storage version
    - Built by RACF Initialization and used when accessing profiles
    - **Could only be updated via IPL prior to z/OS V1R5**

Template Issues prior to z/OS V1R5

- Install a new release or PTF with template changes. If IRRMIN00 not run
  - **Re-IPL** required.
- IRRMIN00 requires correct IRRTEMP1 source. Latest level not obvious.

```
$/VERSION HRF7707
$/VERSION OA01234
```

  - If wrong level used **Re-IPL** required
- Apply a PTF with template changes
  - **Re-IPL** required even if no modifications in PTF require IPL
- Could mistakenly run IRRMIN00 to initialize the active database rather than update it, **wiping out database.**

Dynamic Template Support

- RACF database templates:
  - No longer shipped in source format as IRRTEMP1
  - Shipped as a module in compiled format as IRRTEMP2
  - Contain numeric release and APAR levels so latest level is obvious

$/VERSION FMID/APAR# rrrrrrrr.aaaaaaaa

```
$/VERSION HRF7708 00000010.00000000
$/VERSION OA01234 00000010.00000010
$/VERSION OA01567 00000010.00000020
$/VERSION HRF7709 00000023.00000020
```

- SET LIST operator command displays the in-storage template level and the dynamic parse level in effect on the system.

```
RACF STATUS INFORMATION:
     TEMPLATE VERSION           - HRF7708 00000010.00000000
     DYNAMIC PARSE VERSION      - HRF7708
```

RACF Initialization

During IPL, RACF Initialization puts the highest level of templates in storage *automatically*

- If the Master Primary database template level is higher or the same as RACF Initialization template level, it builds them from the database
- Otherwise, it builds the in-storage templates from RACF initialization template level
  - Warning message issued

```
ICH579E RACF TEMPLATES ON DATABASE ARE DOWNLEVEL:
HRF7708 00000000.00000000; USING TEMPLATES AT LEVEL
HRF7708 00000010.00000000 FROM IRRTEMP2.
RUN IRRMIN00 PARM=UPDATE
```

- **Re-IPL** is not necessary!

IRRMIN00 Enhancements – Database Initialization

- No longer uses the SYSTEMP data set, which typically pointed to SYS1.MODGEN(IRRTEMP1)
  - Instead uses templates in IRRMIN00 load module
- Fails PARM=NEW if the output database is active on the system where IRRMIN00 is invoked
- Will not apply downlevel templates to a database
- Can make templates active dynamically when templates on the active master primary database are a higher level than in-storage templates
  - New PARM=ACTIVATE parameter

IRRMIN00 Parameters

- PARM=NEW
  - Formats a non-VSAM DASD data set as a RACF database
    - Fails if invoked against an active database on the system where IRRMIN00 is invoked
- PARM=UPDATE
  - Writes new templates to the database
    - Fails if new templates are not at higher level than ones in database
- PARM=ACTIVATE
  - If the active master primary database has higher level templates than those in storage, they are copied to storage

Dynamic Templates Summary

No IPL necessary to update RACF database templates

- RACF Initialization builds the in-storage templates automatically from the latest level
- IRRMIN00 PARM=NEW and PARM=UPDATE automatically writes the latest level of templates to the database
- IRRMIN00 PARM=UPDATE will not down-level the templates on the database
- IRRMIN00 PARM=ACTIVATE will dynamically ‘activate’ new templates by replacing the in-storage templates with the new templates
- IRRMIN00 PARM=NEW will not allow an existing, active database to be newly initialized (from the system on which the database is active)

Session Summary

More updates in RACF can be made without an IPL!

- RACF Dynamic Class Descriptor Table
  - How to update the RACF CDT without an IPL

V1R6

- RACF Router Table Enhancements
  - Very few reasons to update it

V1R6

- RACF Dynamic Templates
  - How to update RACF Templates without an IPL

V1R5

Questions???

Appendix

- Websites
  - RACF website:
    http://www.ibm.com/servers/eserver/zseries/zos/racf/
  - RACF downloads (for CDT2DYN EXEC):
    http://www.ibm.com/servers/eserver/zseries/zos/racf/goodies.html
- Publications References
  - SA22-7683 Security Server RACF Security Administrator's Guide
    - Entire chapter on Dynamic CDT
  - SA22-7687 Security Server RACF Command Language Reference
    - CDTINFO keyword on RDEFINE, RALTER, RLIST commands
  - SA22-7687 Security Server RACF RACROUTE Macro Reference
    - NEXT= keyword on RACROUTE REQUEST=STAT
    - Coding example to retrieve all classes in CDT
  - SC26-993 DB2 Universal Database for OS/390® and z/OS Administration Guide

Terminology

CDT: Abbreviation for Class Descriptor Table.

Class descriptor table: A table in RACF which defines general resource class names and attributes. When this term is used without a preceding 'dynamic' or 'static', it refers to a combination of the static class descriptor table and the dynamic class descriptor table, if it exists.

Dynamic CDT: An optional dynamic portion of the RACF class descriptor table which contains class entries built from the CDT general resource class. It does not contain class entries from the IBM-supplied class descriptor table (ICHRRCDX) nor class entries from the installation-defined class descriptor table (ICHRRCDE, if it exists). The dynamic CDT is treated as a logical extension of the static CDT.

IBM-supplied CDT: The class descriptor table that IBM ships with the RACF product. The module shipped is ICHRRCDX. The classes defined here are not to be modified except by IBM.

Installation-defined CDT: An optional additional portion of the CDT which is customized by an installation. The module name is ICHRRCDE. The function provided by this module can be replaced with the dynamic CDT function.

Static CDT: The (non-dynamic) portion of the class descriptor table that is coded in the IBM-supplied CDT (ICHRRCDX) and the Installation-defined CDT (ICHRRCDE).