Jon Furminger – z/OS LDAP Development - furming@us.ibm.com
NYSPA
2 March 2010

Accessing RACF data via the IBM Tivoli Directory Server (IBM TDS) for z/OS

Disclaimer

The information contained in this document has not been submitted to any formal IBM test and is distributed on an "as-is" basis without any warranty either express or implied. The use of this information or the implementation of any of these techniques is a customer responsibility and depends on the customer's ability to evaluate and integrate them into the operational environment. While each item may have been reviewed by IBM for accuracy in a specific situation, there is no guarantee that the same or similar results will be obtained elsewhere. Customers attempting to adapt these techniques to their own environment do so at their own risk.

In this document, any references made to an IBM licensed program are not intended to state or imply that only IBM's licensed program may be used; any functionally equivalent program may be used instead.

Any performance data contained in this document was determined in a controlled environment and therefore, the results which may be obtained in other operating environments may vary significantly.

Users of this document should verify the applicable data for their specific environments. It is possible that this material may contain references to, or information about, IBM products (machines and programs), programming, or services that are not announced in your country or not yet announced by IBM. Such references or information should not be construed to mean that IBM intends to announce such IBM products, programming, or services.

Permission is hereby granted to publish an exact copy of this paper in the Solutions proceedings. IBM retains the title to the copyright in this paper, as well as the copyright in all underlying works. I BM retains the right to make derivative works and to republish and distribute this paper to whomever it chooses in any way it chooses.

Trademarks

- See url http://www.ibm.com/legal/copytrade.shtml for a list of trademarks.

Agenda

- LDAP Overview
- IBM Tivoli Directory Server (TDS) for z/OS Overview
- Using the SDBM (RACF) Backend
- IBM TDS for z/OS Authentication Mechanisms
- Changing RACF Password or Password Phrase
- LDAP-RACF Change Logging
- Remote Authorization and Audit Services
- Conclusion

LDAP Overview

What is LDAP?

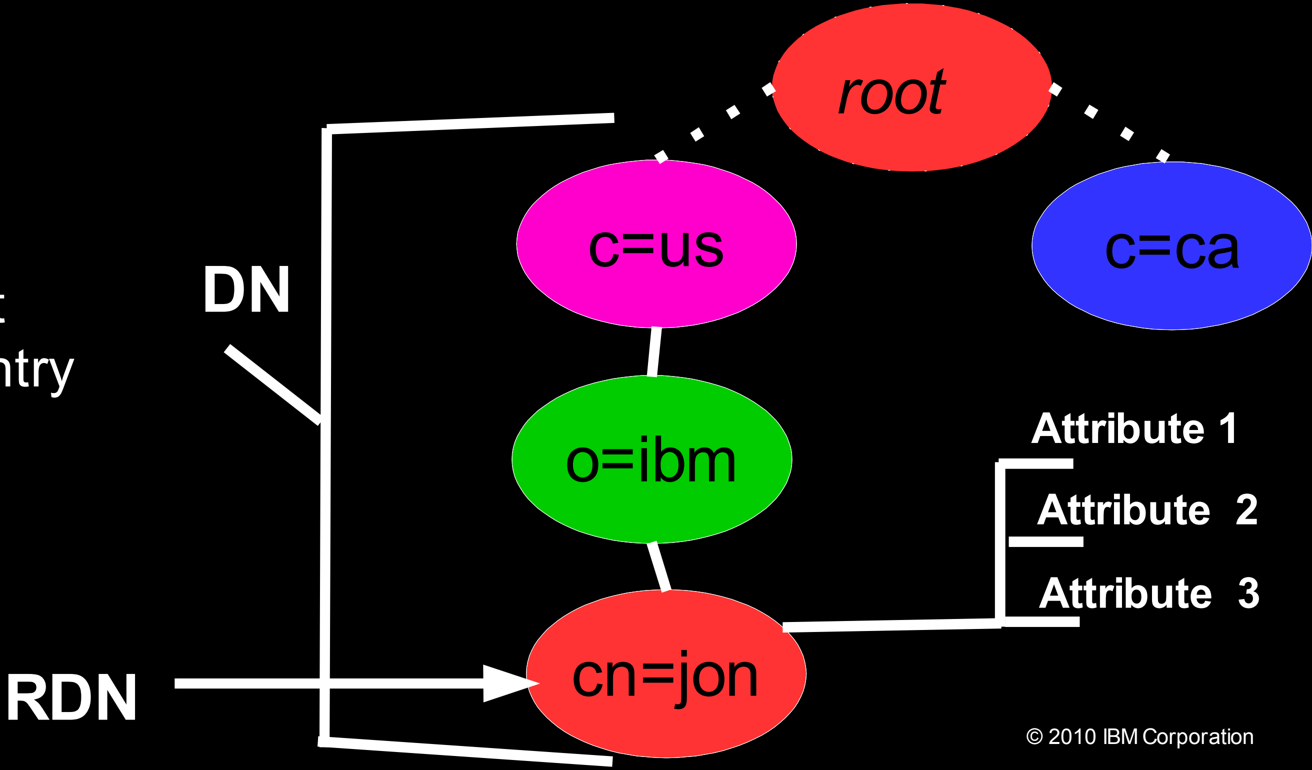
- LDAP – Lightweight Directory Access Protocol
  - Originally developed as front-end of X.500 (DAP)
  - TCP/IP based wire protocol for updating directory information
  - Industry standard protocol defined in IETF RFCs
    - Servers and clients reside on different platforms
    - Allows adding, modifying, deleting, searching, and comparing entries in a directory
  - Optimized for searching vs. adding or modifying
  - Commonly used for authentication

- What is a directory?
  - Directory model is based on entries
    - Each entry is identified by a distinguished name (DN)
    - DN: cn=jon,o=ibm,c=us

What is LDAP? (continued)

- Each entry is a collection of attributes
  - Each attribute has a type and values
  - Attributes are grouped into object classes (determine optional and required attributes)
  - Schema defines attributes and object classes

```
dn: cn=jon,o=ibm,c=us
objectclass: person
cn: jon
sn: cottrell
userpassword: mysecret
description: A sample entry
```



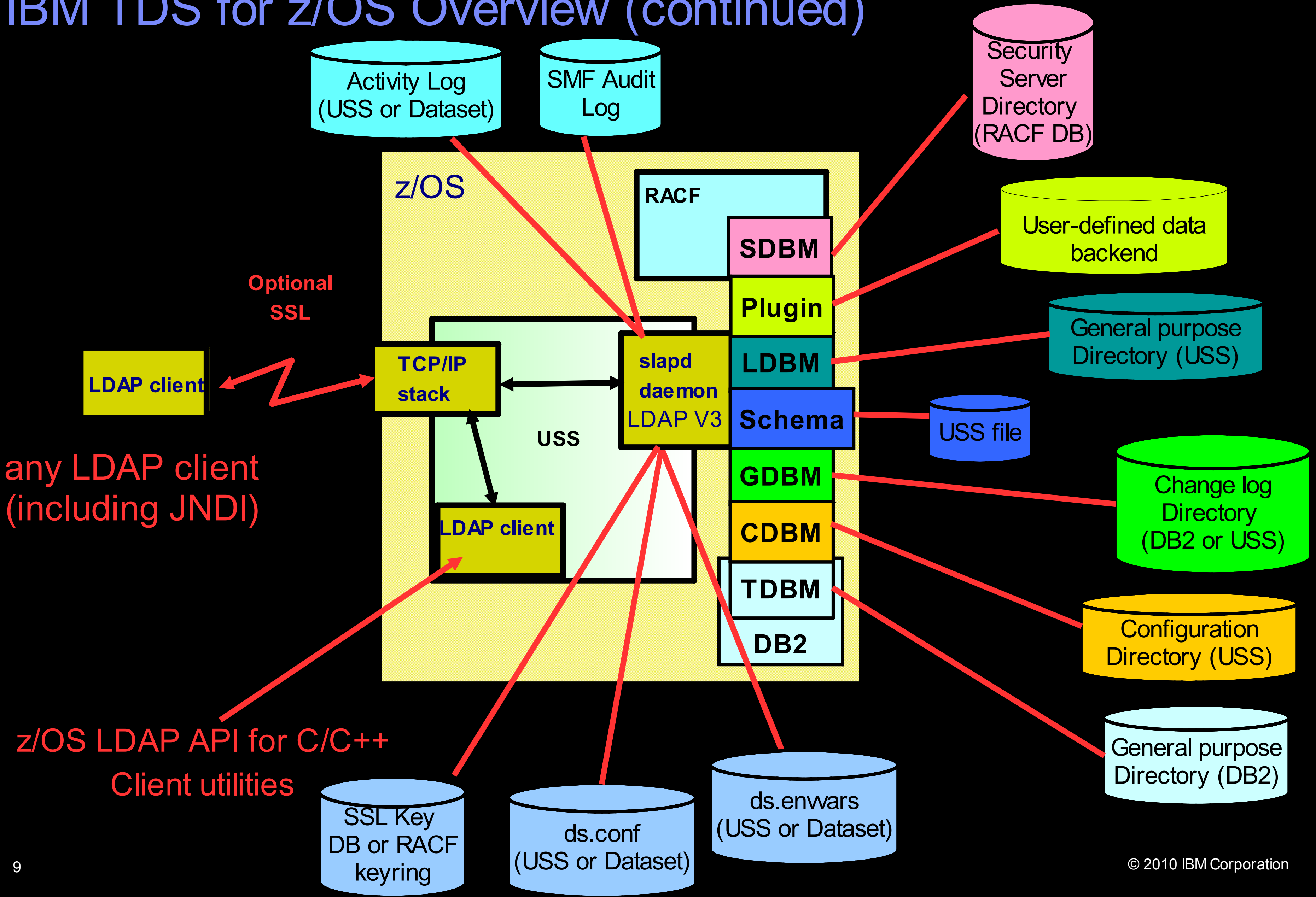
IBM TDS for z/OS Overview

- IBM TDS available since z/OS R8
  - Free product in base z/OS
  - Previous product on z/OS called Integrated Security Services (ISS) LDAP server
  - ISS is no longer shipped in z/OS R11

- Server runs in 31 or 64 bit mode as an APF-authorized program

- Common LDAP operations (add, compare, delete, search, modify) are provided by client utilities in TSO and USS:
  - ldapadd, ldapcompare, ldapdelete, ldapmodify, ldapmodrdn, ldapsearch

IBM TDS for z/OS Overview (continued)


Activity Log (USS or Dataset)
SMF Audit Log
Security Server Directory (RACF DB)
z/OS
RACF
SDBM
User-defined data backend
Plugin
General purpose Directory (USS)
LDBM
Optional SSL
LDAP client
TCP/IP stack
slapd daemon LDAP V3
USS
Schema
USS file
GDBM
Change log Directory (DB2 or USS)
CDBM
LDAP client
TDBM
DB2
Configuration Directory (USS)
General purpose Directory (DB2)
SSL Key DB or RACF keyring
ds.conf (USS or Dataset)
ds.envvars (USS or Dataset)

any LDAP client (including JNDI)

z/OS LDAP API for C/C++
Client utilities

Using the SDBM (RACF) Backend

SDBM Backend Overview

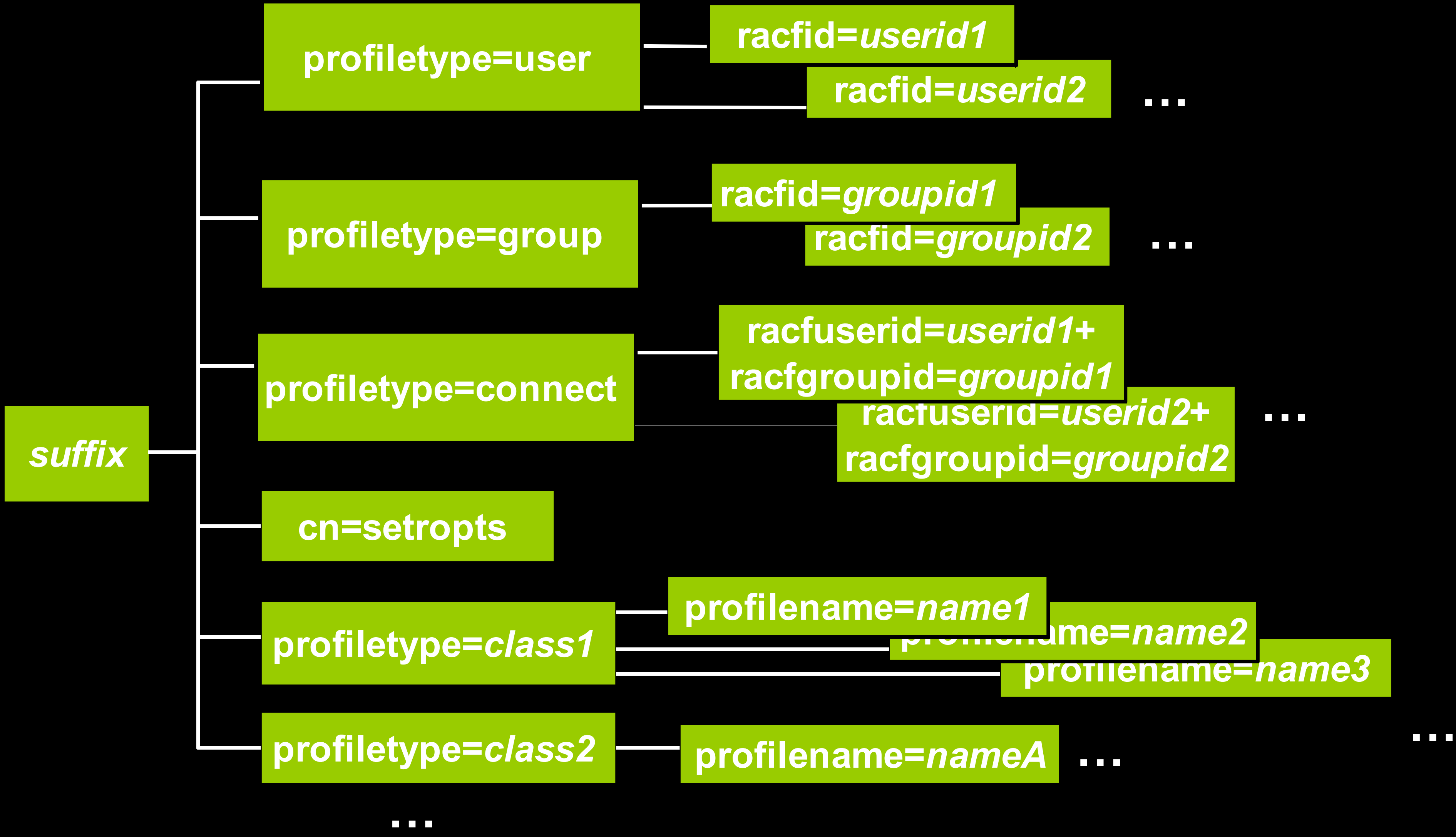
- Provides these features remotely via LDAP protocol:
  - Authentication with users
  - Add, modify, delete RACF users, groups, and general resources
  - Add, modify, and delete user connections to groups
  - Add and remove users and groups in general resource profiles
  - Modify SETROPTS options that affect classes
  - Retrieve RACF information for users, groups, connections, general resources, and class options
  - Retrieve RACF user password and password phrase envelopes

SDBM Backend Overview (continued)

- Converts LDAP operations into RACF commands and services
  - An LDAP add of a RACF user is converted to an ADDUSER command and issued by R_Admin
  - An LDAP search of a RACF resource profile is converted into an R_Admin profile extract
- RACF commands are issued under the bound user's authority
  - Via LDAP you cannot do anything that TSO does not allow
- TDS for z/OS does not copy the data out of the RACF DB
- SDBM configuration is simple, update LDAP configuration file:

```
database sdbm GLDBSD31/GLDBSD64
suffix cn=sdbm
enableResources on
```

SDBM Backend Directory Hierarchy



suffix

profiletype=user

racfid=*userid1*

racfid=*userid2*

…

profiletype=group

racfid=*groupid1*

racfid=*groupid2*

…

profiletype=connect

racfuserid=*userid1*+
racfgroupid=*groupid1*

racfuserid=*userid2*+
racfgroupid=*groupid2*

…

cn=setropts

profiletype=*class1*

profilename=*name1*

profilename=*name2*

profilename=*name3*

…

profiletype=*class2*

profilename=*nameA*

…

…

Example DN: racfid=jon,profiletype=user,cn=sdbm

SDBM Schema

- SDBM distinguished names (DNs):
  - User: racfid=jon,profiletype=user,cn=sdbm
  - Group: racfid=groupc,profiletype=group,cn=sdbm
  - User-Group connection: racfuserid=jon +racfgroupid=groupc,profiletype=connection,cn=sdbm
  - Resource profile: profilename=TERM1,profiletype=TERMINAL,cn=sdbm
  - Setropts: cn=setropts,cn=sdbm
- Initial (minimum) LDAP schema is sufficient for RACF fixed fields
  - Each RACF add/alt/listuser, add/alt/listgrp, connect, rdefine,ralter,rlist keyword is mapped to an LDAP attribute
    - OMVS uid keyword <--> racfOmvsUid attribute

Using SDBM – Examples

- Add a RACF user entry
  - Create a file, u1234.ldif, containing an entry to be added:

```
dn: racfid=u1234,profiletype=user,cn=sdbm
objectclass: racfUser
objectclass: racfUserOmvsSegment
racfid: u1234
racfdefaultgroup: group1
racfowner: radmin
racfattributes: special
racfomvsuid: 1234
racfomvshome: /home/u1234
```

  - Invoke the ldapadd utility:
    - ldapadd -D “racfid=radmin,profiletype=user,cn=sdbm” -w radminpw -f u1234.ldif
  - SDBM executes under the context of bound (radmin) user:
    - ADDUSER u1234 OWNER(radmin) DFLTGRP(group1) SPECIAL OMVS(UID(1234) HOME(/home/u1234))

Using SDBM – Examples (continued)

- Modifying a RACF user entry
  - Create a file, modu1234.ldif, containing the modification:

```
dn: racfid=u1234,profiletype=user,cn=sdbm
changetype: modify
add: racfBuilding
racfBuilding: 256
-
add: racfDepartment
racfDepartment: LDAP
```

  - Invoke the ldapmodify utility:
    - ldapmodify -D “racfid=radmin,profiletype=user,cn=sdbm” -w radminpw -f modu1234.ldif
  - SDBM executes under the context of bound (radmin) user:
    - ALTUSER U1234 WORKATTR(WABLDG('256') WADEPT('LDAP'))

Using SDBM – Examples (continued)

- Display a RACF user-group connection:
  - Invoke the ldapsearch utility:
    - ldapsearch -L -D “racfid=radmin,profiletype=user,cn=sdbm” -w radminpw -b “racfuserid=u1234 +racfgroupid=group1,profiletype=connect,cn=sdbm” “objectclass=*”

    - SDBM executes under the context of bound (radmin) user: LISTUSER U1234 and returns group info for GROUP1

```
dn: racfuserid=U1234+racfgroupid=GROUP1,profiletype=CONNECT,cn=sdbm
racfuserid: U1234
racfgroupid: GROUP1
racfconnectauthdate: 02/08/10
racfconnectowner: RACFID=RADMIN,PROFILETYPE=USER,CN=SDBM
racfconnectgroupauthority: USE
racfconnectgroupuacc: NONE
racfconnectcount: 0
objectclass: TOP
objectclass: RACFBASECOMMON
objectclass: RACFCONNECT
```

Using SDBM – Examples (continued)

- Add a RACF resource profile to the FACILITY class
  - – Create file, mine.ldif, containing an entry to be added:

```
dn: profilename=TERM1,profiletype=TERMINAL,cn=sdbm
objectclass: racfresource
racfOwner: GROUP1
racfUacc: NONE
racfaccesscontrol: ID(U2) ACCESS(READ)
```

  - – Invoke the ldapadd utility:
    - ldapadd -D “racfid=radmin,profiletype=user,cn=sdbm” -w radminpw -f mine.ldif
  - – SDBM executes under the context of bound (radmin) user:
    - RDEFINE TERMINAL TERM1 OWNER(GROUP1) UACC(NONE)
    - PERMIT TERM1 CLASS(TERMINAL) ID(U2) ACCESS (READ)

Using SDBM – Examples (continued)

- Refresh the FACILITY class
  - Create file, refresh.ldif, containing the modification to the cn=setropts entry:

```
dn: cn=setropts,cn=sdbm
changetype: modify
replace: racfsetroptsattributes
racfsetroptsattributes: REFRESH
-
replace: racfraclist
racfraclist: profiletype=FACILITY,cn=sdbm
```

  - Invoke the ldapmodify utility:
    - ldapmodify -D “racfid=radmin,profiletype=user,cn=sdbm” -w radminpw -f refresh.ldif
  - SDBM executes under the context of bound (radmin) user:
    - SETROPTS REFRESH RACLIST(FACILITY)

RACF (SDBM) Custom Fields

- Create an LDAP attribute to map the RACF PHONE field in the USER CSDATA segment
- ldapmodify -D *adminDn* -w *adminPw* -f schema.mod

```
dn: cn=schema
changetype: modify
add: attributetypes
attributetypes: (
 racfphone-OID
 NAME 'racfphone'
 DESC 'Represents the PHONE field in the RACF user CSDATA segment'
 EQUALITY caseIgnoreMatch
 SYNTAX 1.3.6.1.4.1.1466.115.121.1.26
 SINGLE-VALUE
 USAGE userApplications
 )
-
add: ibmattributetypes
ibmattributetypes: (
 racfphone-OID
 ACCESS-CLASS sensitive
 RACFFIELD ('USER-CSDATA-PHONE' 'char')
 )
```

RACF (SDBM) Custom Fields (continued)

- Modify RACF user, u1234, to add the racfphone attribute

- Create file, modu1234.ldif, to contain the modification:

```
dn: racfid=u1234,profiletype=user,cn=sdbm
changetype: modify
add: racfphone
racfphone: 123-456-7890
```

- Invoke the ldapmodify utility
  - ldapmodify -D “racfid=radmin,profiletype=user,cn=sdbm” -w radmin -f modu1234.ldif

- SDBM executes under the context of bound (radmin) user:
  - ALTUSER U1234 CSDATA(PHONE(123-456-7890))

IBM TDS for z/OS Authentication Methods

IBM TDS for z/OS Authentication Mechanisms

- LDAP is a “stateful” protocol
  - Session starts when client binds to server
    - Can be encrypted with SSL to protect data during transmission
  - Authentication is performed during bind
    - Check password or certificate
    - Determine groups to which user belongs (for authorization checking)

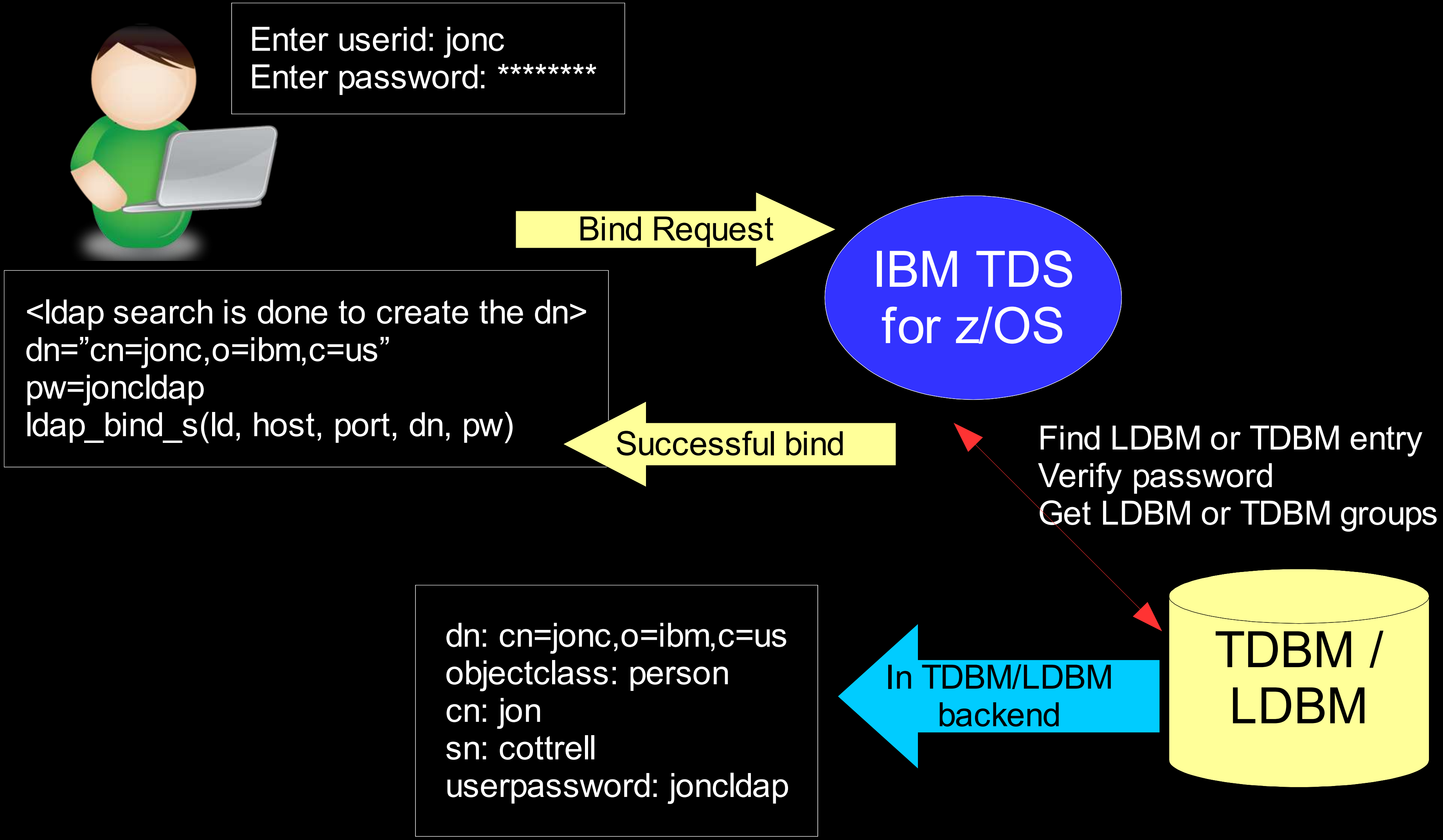
- Simple bind: Distinguished name and password
  - Passwords can be stored in the following locations:
    - TDBM or LDBM – Hashed with crypt, MD5 or SHA-1 or two-way encryption with AES or 3DES
    - RACF

IBM TDS for z/OS Authentication Mechanisms (continued)

- EXTERNAL bind: X.509 certificate over SSL
  - Distinguished name in certificate is used as authorization DN
  - Certificates can be mapped to a RACF user ID
    - Use the RACDCERT MAP command to create mapping

- GSSAPI (Kerberos) bind: Kerberos principal sends ticket for LDAP server
  - Kerberos principal can be mapped to RACF, TDBM, and LDBM user

TDBM and LDBM Simple Authentication



Enter userid: jonc
Enter password: ********

Bind Request

IBM TDS for z/OS

<ldap search is done to create the dn>
dn=”cn=jonc,o=ibm,c=us”
pw=joncldap
ldap_bind_s(ld, host, port, dn, pw)

Successful bind

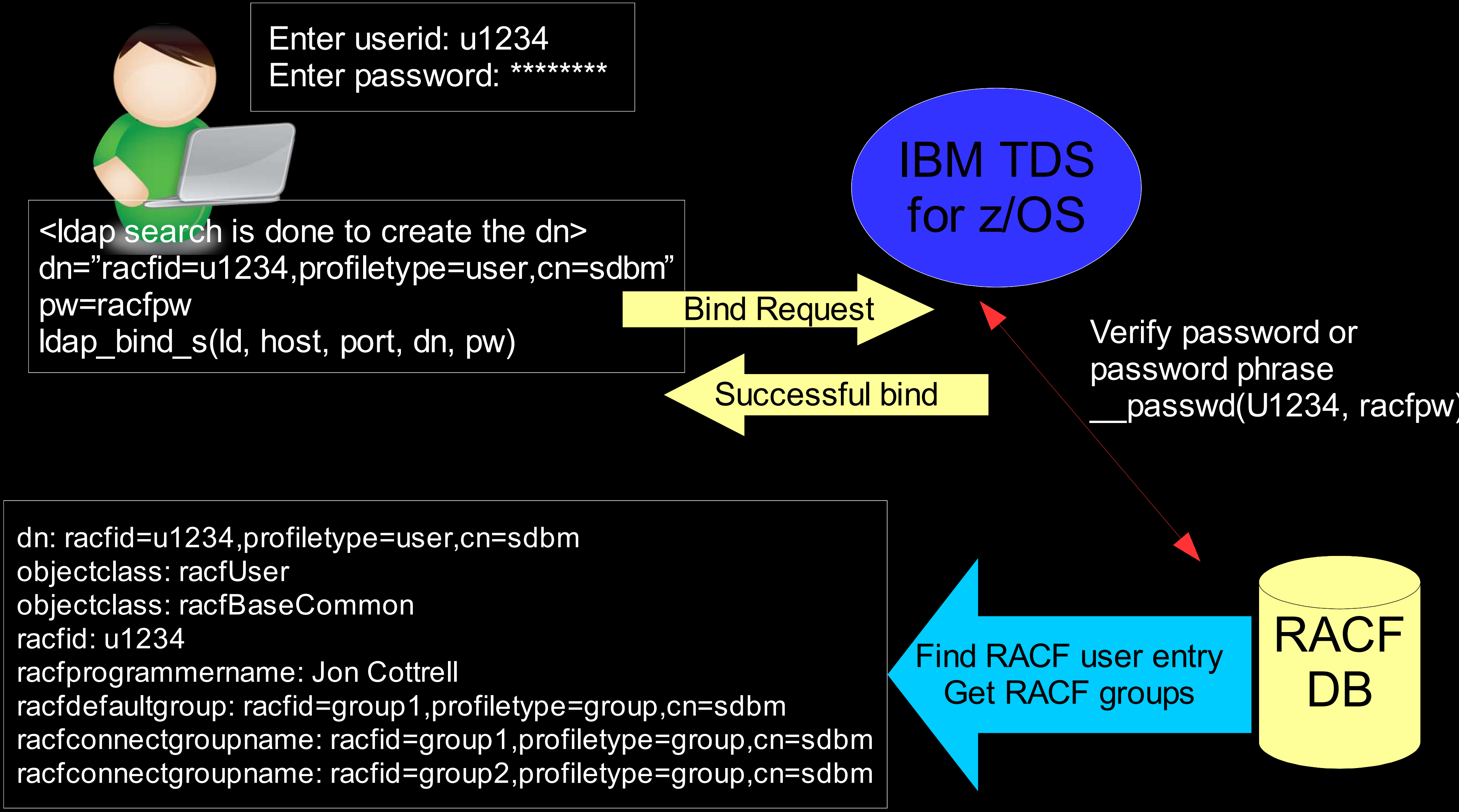
Find LDBM or TDBM entry
Verify password
Get LDBM or TDBM groups

dn: cn=jonc,o=ibm,c=us
objectclass: person
cn: jon
sn: cottrell
userpassword: joncldap

In TDBM/LDBM backend

TDBM / LDBM

SDBM (RACF) Simple Authentication



Enter userid: u1234
Enter password: ********

<ldap search is done to create the dn>
dn=”racfid=u1234,profiletype=user,cn=sdbm”
pw=racfpw
ldap_bind_s(ld, host, port, dn, pw)

Bind Request

Successful bind

IBM TDS for z/OS

Verify password or
password phrase
__passwd(U1234, racfpw)

RACF DB

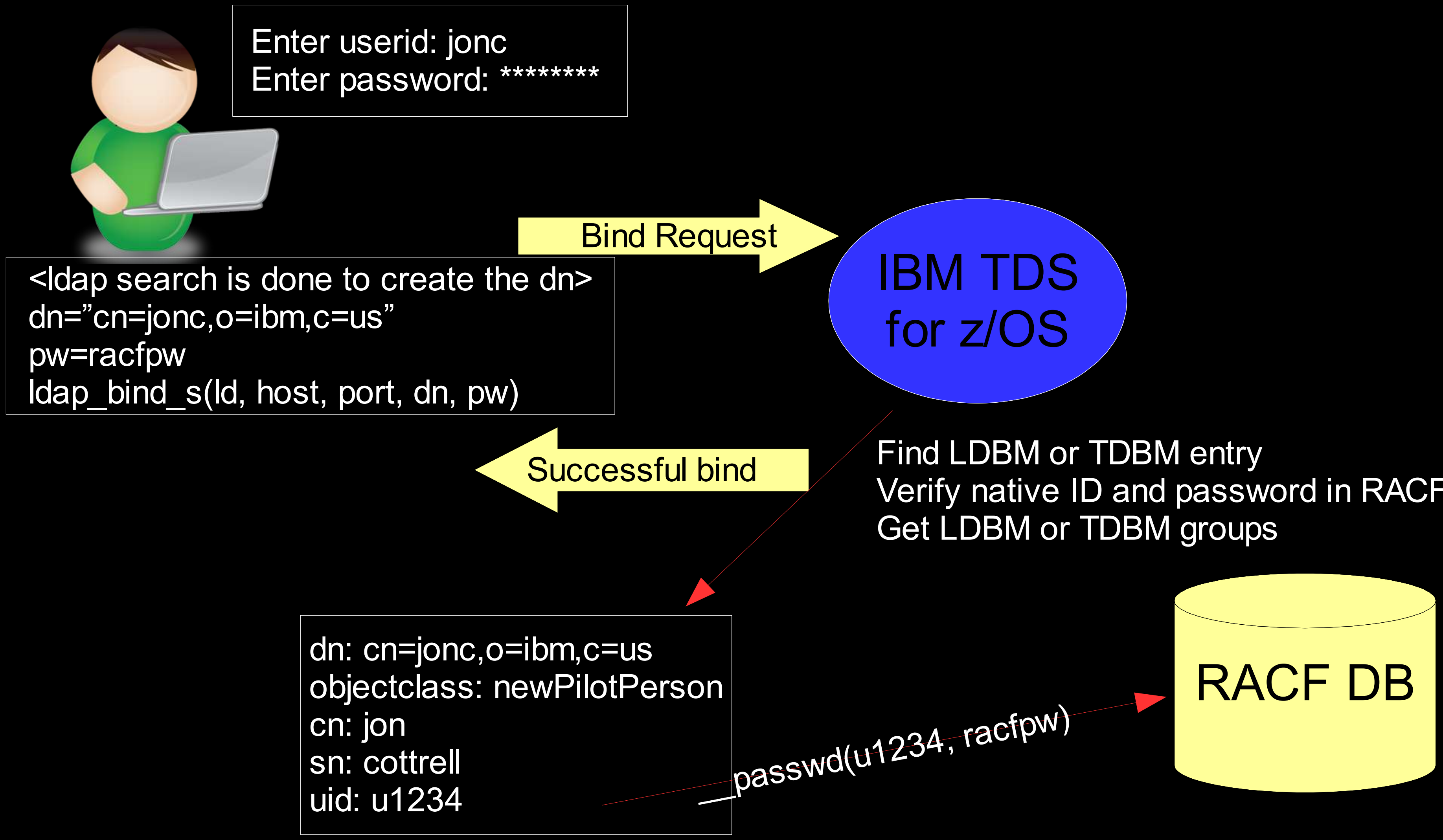
Find RACF user entry
Get RACF groups

```
dn: racfid=u1234,profiletype=user,cn=sdbm
objectclass: racfUser
objectclass: racfBaseCommon
racfid: u1234
racfprogrammername: Jon Cottrell
racfdefaultgroup: racfid=group1,profiletype=group,cn=sdbm
racfconnectgroupname: racfid=group1,profiletype=group,cn=sdbm
racfconnectgroupname: racfid=group2,profiletype=group,cn=sdbm
```

TDBM and LDBM Native Authentication

- Disadvantages of authentication in TDBM and LDBM
  - Another password repository to manage because password stored in the TDBM or LDBM entry
- Disadvantages of authentication in RACF
  - SDBM backend required with long DNs
  - Non-standard schema: Only supported for RACF
  - Limited search capabilities
- Native authentication – Uses entries in TDBM or LDBM but password or password phrase is stored in RACF
  - Standard distinguished names (e.g. cn, o, c)
  - Authentication (password verification) performed by RACF
    - No need for administration or synchronization of multiple password registries
    - RACF authentication triggered by **uid** or **ibm-nativeId** attribute

TDBM and LDBM Native Authentication (continued)



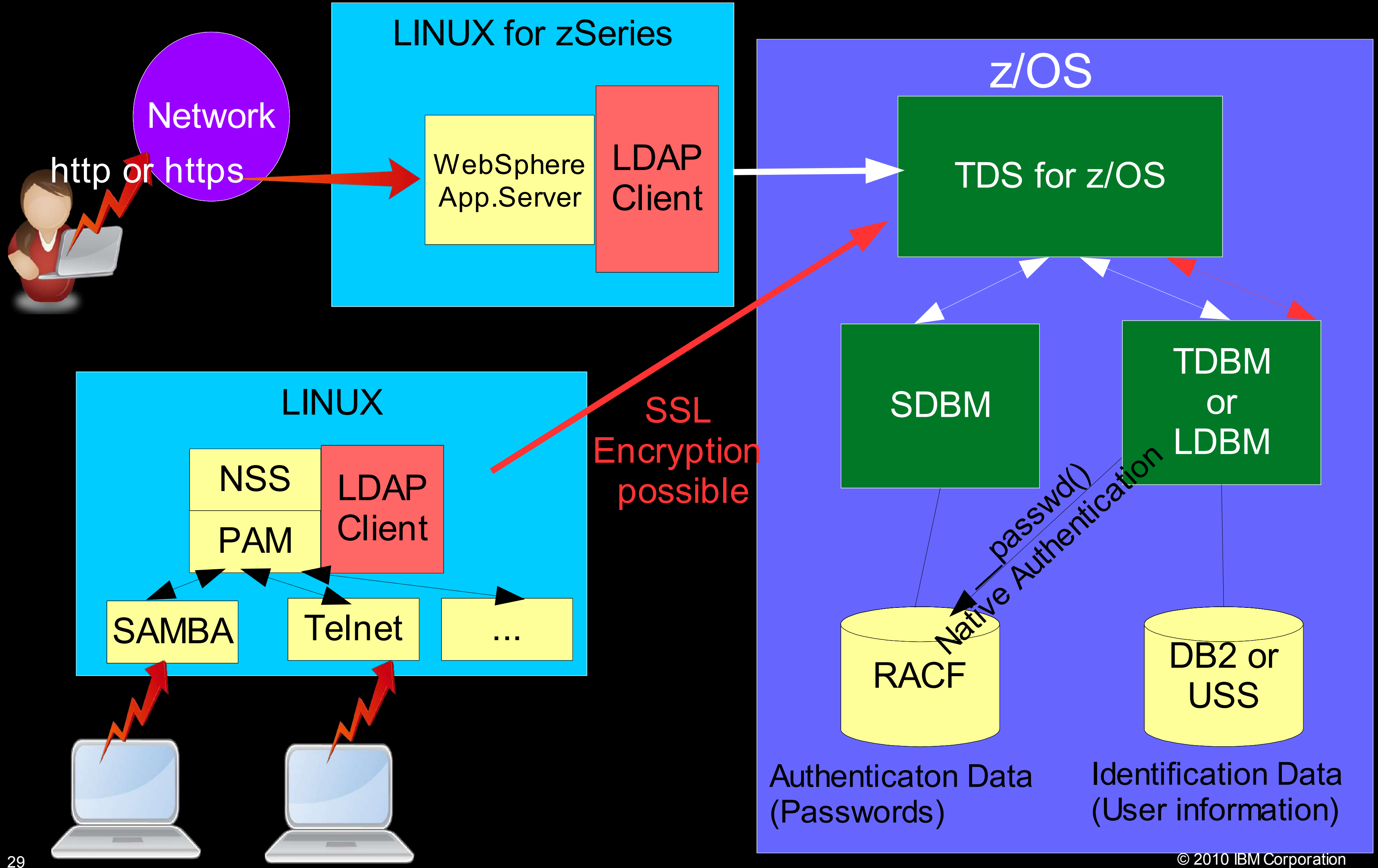
Enter userid: jonc
Enter password: ********

<ldap search is done to create the dn>
dn=”cn=jonc,o=ibm,c=us”
pw=racfpw
ldap_bind_s(ld, host, port, dn, pw)

Find LDBM or TDBM entry
Verify native ID and password in RACF
Get LDBM or TDBM groups

dn: cn=jonc,o=ibm,c=us
objectclass: newPilotPerson
cn: jon
sn: cottrell
uid: u1234

Using LINUX to Authenticate to TDS



Changing RACF Password
or
Password Phrase

Changing RACF Password or Password Phrase

- The ldapmodify utility can be used to change RACF password or password phrase
  - Via SDBM backend:

```
dn: racfid=u1234,profiletype=user,cn=sdbm
replace: racfPassword
racfPassword: mynewpw
racfAttributes: noexpired
```

  - Via LDBM or TDBM with native authentication:

```
dn: cn=jon,o=ibm,c=us
delete: userPassword
userPassword: racfpw
-
add: userPassword
userPassword: mynewpw
```

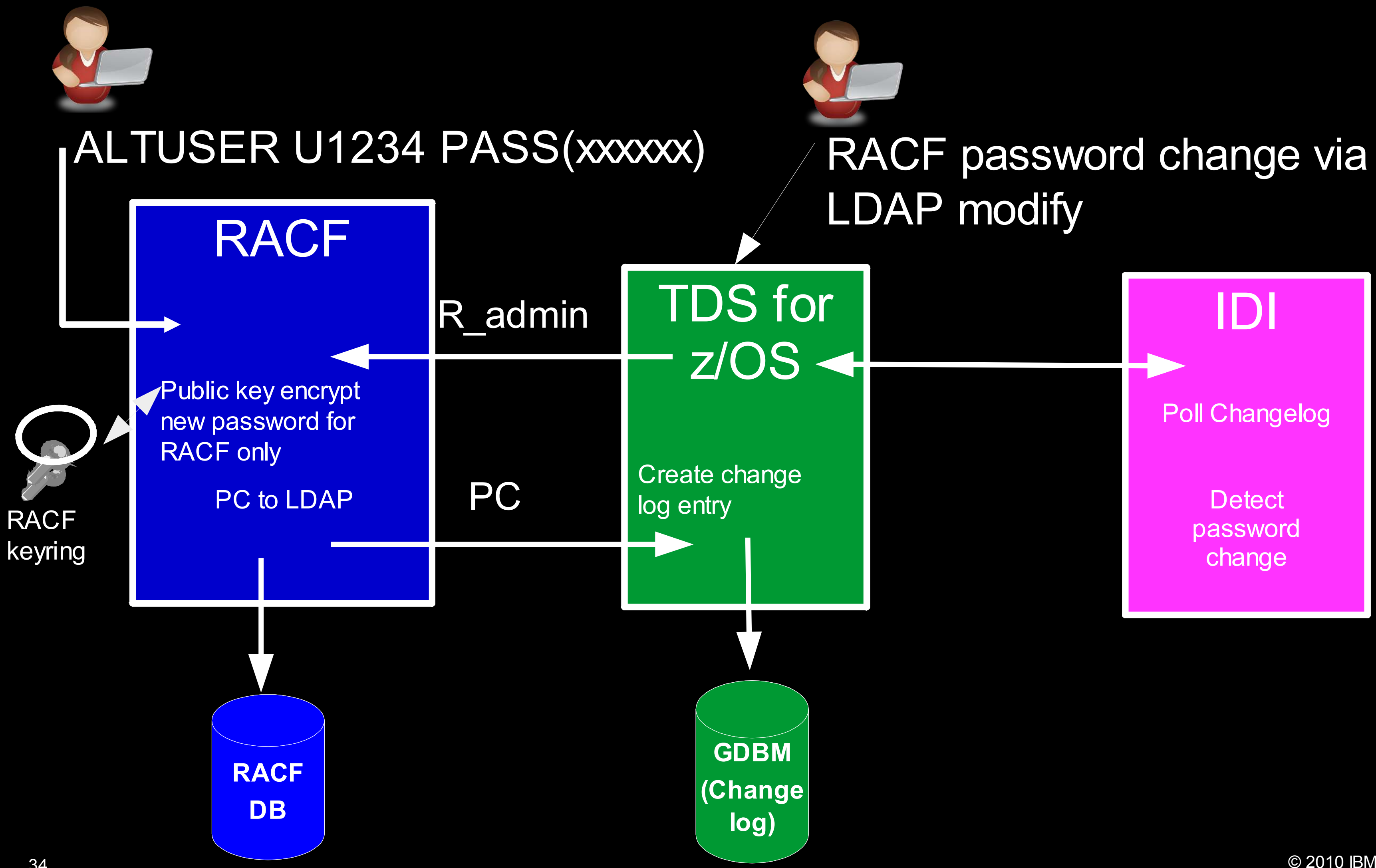
    - Note: replace: userPassword is not supported when changing the RACF password with native authentication

Changing RACF Password or Password Phrase (continued)

- SDBM or native authentication bind can be used to change a password (even if expired)
  - Specify *old_password/new_password* as password value when authenticating
    - ldapsearch -D “racfid=u1234,profiletype=user,cn=sdbm” -w mynewpw/new2pass -s base -b “” “objectclass=*”

LDAP-RACF
Change Logging

LDAP-RACF Change Logging



ALTUSER U1234 PASS(xxxxxx)

RACF password change via LDAP modify

RACF

R_admin

TDS for z/OS

IDI

Public key encrypt new password for RACF only

PC to LDAP

PC

Create change log entry

Poll Changelog

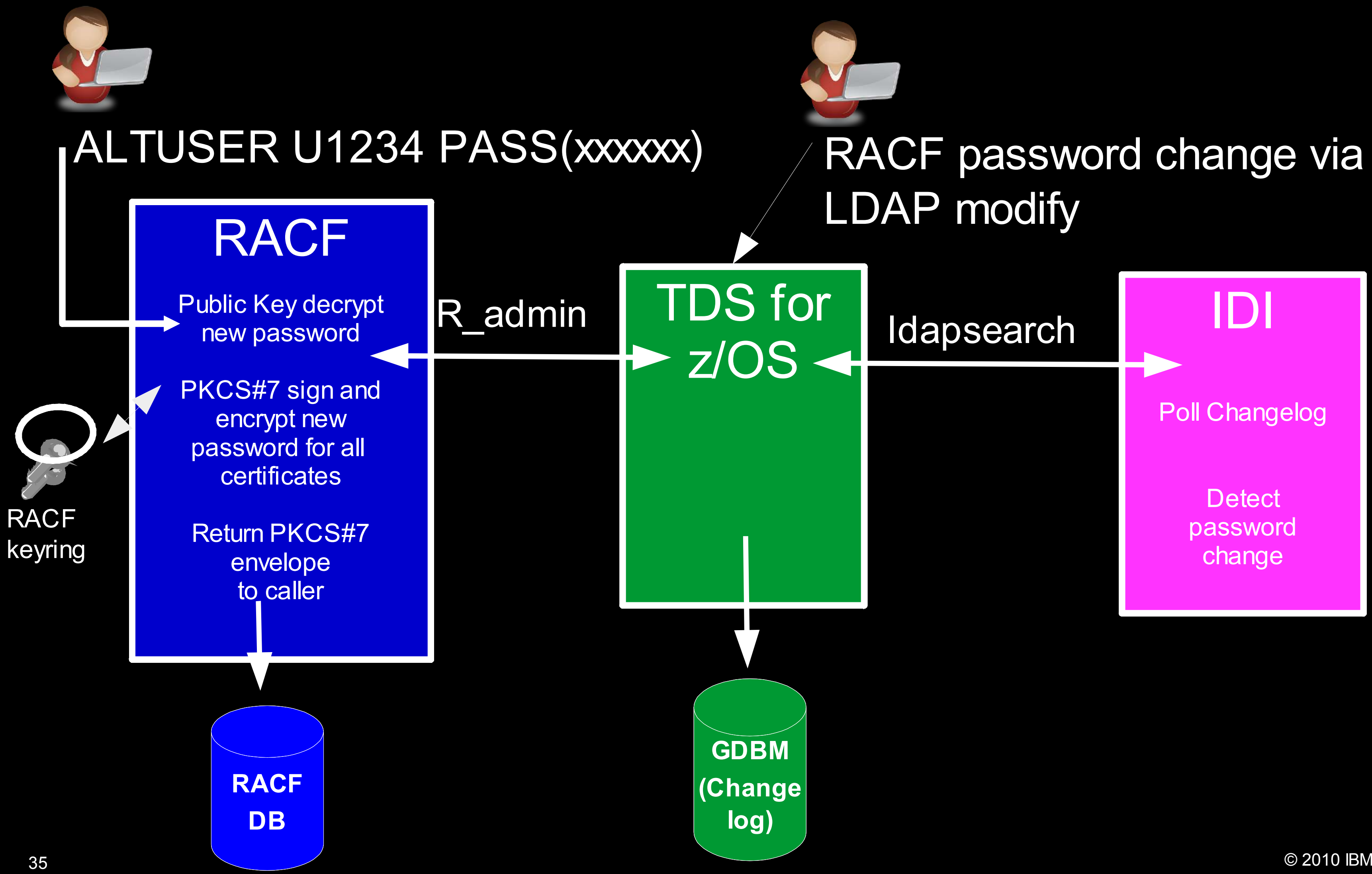
Detect password change

RACF keyring

RACF DB

GDBM (Change log)

LDAP-RACF Change Logging (continued)


ALTUSER U1234 PASS(xxxxxx)
RACF password change via LDAP modify
RACF
Public Key decrypt new password
PKCS#7 sign and encrypt new password for all certificates
Return PKCS#7 envelope to caller
RACF keyring
R_admin
TDS for z/OS
ldapsearch
IDI
Poll Changelog
Detect password change
RACF DB
GDBM (Change log)

LDAP–RACF Change Logging (continued)

- Searching the change log using the ldapsearch utility:
  - ldapsearch -D “racfid=radmin,profiletype=user,cn=sdbm” -w radmin -b “cn=changelog” “changeNumber>= 53829”

```
changeNumber=53289,cn=changelog
objectclass=top
objectclass=changeLogEntry
objectclass=ibm-changeLog
changenumber=53289
changetype=modify
targetdn=RACFID=U1234,PROFILETYPE=USER,CN=SDBM
changes=replace: racfPassword
racfPassword: *ComeAndGetIt*


-
ibm-changeinitiatorsname=RACFID=RADMIN,PROFILETYPE=USER,CN=SDBM
changetime=20100209200313.418178Z
```

LDAP–RACF Change Logging (continued)

- Retrieving RACF envelope containing new password:
  - ldapsearch -L

```
    -D “racfid=radmin,profiletype=user,cn=sdbm”
    -w radmin -b “racfid=u1234,profiletype=user,cn=sdbm”
    “objectclass=*” racfpasswordEnvelope
```

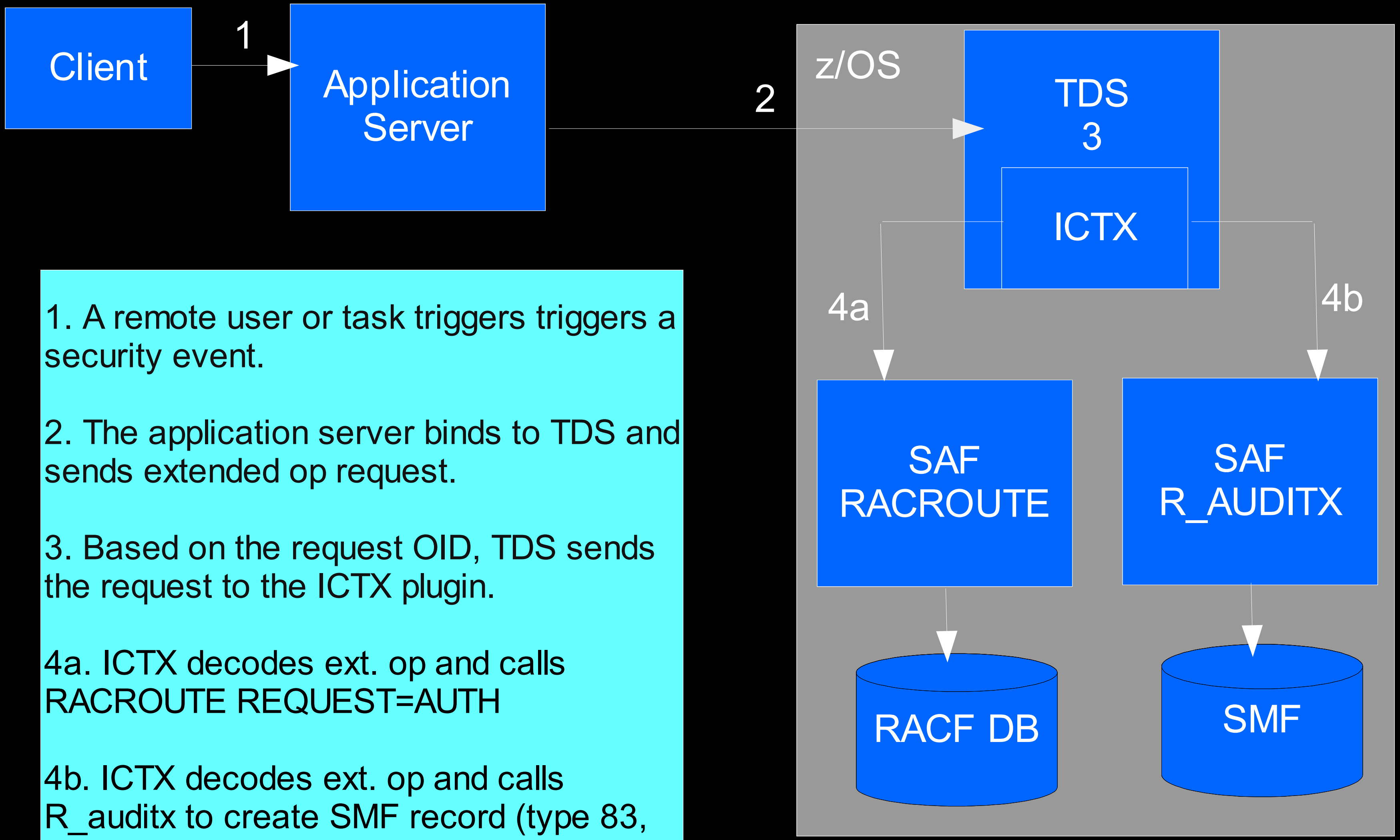
```
dn: racfid=U1234,profiletype=USER,cn=SDBM
racfPasswordEnvelope:: base64_pkcs7_password_envelope
```

Remote Authorization and Audit Services

Remote Authorization and Audit Services

- Two remote services added to enable distributed applications to access security functions on z/OS:
  - Remote Authorization Service – Allows applications to remotely query a z/OS system to check a user’s authority to a resource
    - Can be thought of as a remote interface to the RACROUTE REQUEST-AUTH service
  - Remote Audit Service – Allows applications to remotely write audit records to the z/OS Systems Management Facility (SMF) – Security records (SMF-83)
    - Can be thought of as a remote interface to the R_AUDITX SAF callable service
- These services can be accessed remotely by sending extended operations requests to TDS

Remote Authorization and Audit Services (continued)



1. A remote user or task triggers triggers a security event.

2. The application server binds to TDS and sends extended op request.

3. Based on the request OID, TDS sends the request to the ICTX plugin.

4a. ICTX decodes ext. op and calls RACROUTE REQUEST=AUTH

4b. ICTX decodes ext. op and calls R_auditx to create SMF record (type 83, subtype 4)

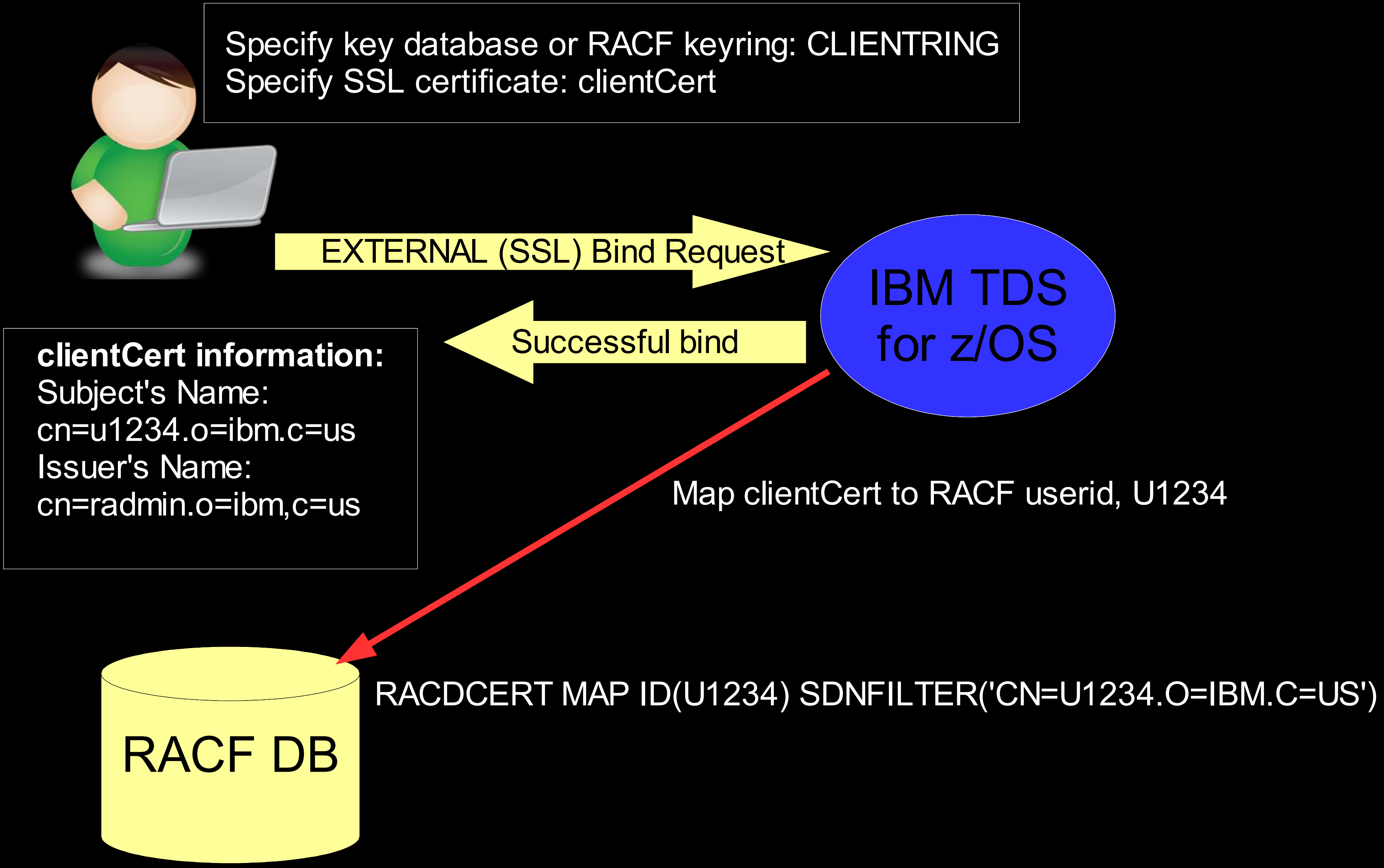
Conclusion

- More information:
  - IBM Tivoli Directory Server Administration And Use for z/OS (SC23-5191)
  - IBM Tivoli Directory Server Client Programming for z/OS (SA23-2214)
  - IBM Tivoli Directory Server Plug-in Reference for z/OS (SA76-0148)

- Contact Information:
  - Jon Furminger
    - Email: furming@us.ibm.com

Appendix
Additional Information

EXTERNAL (SSL Certificate) Mapping



Specify key database or RACF keyring: CLIENTRING
Specify SSL certificate: clientCert

EXTERNAL (SSL) Bind Request

Successful bind

IBM TDS for z/OS

clientCert information:
Subject's Name:
cn=u1234.o=ibm.c=us
Issuer's Name:
cn=radmin.o=ibm,c=us

Map clientCert to RACF userid, U1234

RACF DB

RACDCERT MAP ID(U1234) SDNFILTER('CN=U1234.O=IBM.C=US')