Centralized management for SecureEntry

Before you begin

Before installing UCM, you should read carefully this on-line manual, and be sure that all prerequisites are met. Notice that this is a quite impressive piece of software, but because of its nature, it is very important that you know what you are doing when using it.

Also look for a file named *README.DOC* in the first installation diskette of SecureEntry/2, for last minute changes and detailed driver changes description.

Copyright

Overview

UCM (Users Centralized Management) is an extension to SecureEntry that allows for managing of SecureEntry users and groups in a centralized (corporate) way, with a unique logical view of all your SecureEntry branches. This is done by storing all of SecureEntry information in a centralized site (UCM Host system), in such a way that all of the branch's data becomes effectively a 'shadow' of this central repository.

When a SecureEntry UCM user logs on to one of the UCM administered machines, its security profile information is downloaded, if unknown or modified since last sign-on, and the correct (last) security profiles are activated for him.

As an additional benefit, UCM provides SecureEntry with evident mobility advantages, since any corporate user will be able to sign-on into any of the UCM workstations.

Refer to the SecureEntry administrator's guide for further information about SecureEntry itself.

This on-line manual describes the UCM operation. This includes the installation process, run time procedures, customer file specification and a problem determination guide, which lists the possible messages and return codes.

***** NOTE *****

You will find dataset references in the examples of the guide with the following naming conventions:

UCM.Vxx.

Where **Vxx** identifies the current UCM version.

***** NOTE *****

If you have not contracted the UCM solution, then the only part of this document that may be of any use for you is the return codes one, specifically the one which explains the netbios and EDYSRV/EDYFREE return codes, since this is a common piece used by all LAN based installations.

***** NOTE *****

The UCM code can run either in a MVS/390 or an OS/2 environment. You will find a small table attached to each explained function describing under which target operating system it can run.

Follows a list with the changes made to the documentation of this guide:

(14/02/2000) UCM V4R3. Secure transactions against RACF (EDYURACF).

(15/11/1999) UCM V4R3. UCMLOG and cache for dynamic refresh feature.

(18/08/1999) UCM V4R3. OS/2 Version.

(19/06/1999) UCM V4R3. APPC, Multi-trans transactions.

(28/04/1999) UCM V4R3. RACF emulation.

UCM installation modes

Users Centralized Management (UCM) is a SecureEntry/2 extension that you can use to centralize your users administration tasks. UCM is available under OS/2 and MVS/ESA (OS/390) environments. Follows a list describing the main features of this solution :

1. **Centralized users authentication**

   When you install UCM, SecureEntry/2 users authentication will be managed in a centralized way. You can customize the following security subsystems as your main authentication subsystem within the Host environment:

   RACF (Resource Access Control Facility) *(available under MVS)*

   Security subsystem already installed for your corporation (RACF, TOP SECRET, ACF/2... or any fully compatible replacement). SecureEntry/2 - UCM can access RACF in two different ways:

   DIRECT

   SecureEntry/2 uses a standard SNA Service TP to access RACF. Unless you have SNA level encryption installed, this mode is subject to line peeking by unauthorized third parties.

   SCRAMBLED (EDYURACF)

   You can use the UCM data scrambling feature to send the user data scrambled through the communications channel to RACF. SecureEntry will then access RACF through the UCM provided EDYURACF transaction program in the host.

UCM RACF EMULATOR *(available under OS/2 or MVS environments)*

You can use UCM as your security subsystem to authenticate SecureEntry/2 users at connection time instead of RACF. This basically means that the UCM repository will be used also to store the different user's passwords. When using this authentication subsystem, UCM data scrambling will always be used. The transaction program that will be used then is the UCM provided EDYTP TP.

2. **Central management of security policies**

With UCM, you can define the SecureEntry/2 security policies for your users, groups and network resources either :

Through UCM (centralized management)

Through UCM, you can manage in a centralized way all the corporate security policies for your users, groups and network resources through SecureEntry administration tools, working in centralized mode. This policies and definitions will reside in the corporate UCM repository (DB2 Database), and transaction processes will be used to distribute the users data to the branches, at sign-on time, either with:

Standard TP's, or

Multi-trans TP's *(only available under MVS)* for improved performance.

Besides, and for branch data (non user specific), you can decide between the following branch refresh methods :

ONLINE

Branch data will be downloaded to your corporate branches by UCM at a preconfigured time (either at logon time, at a fixed time, or startup time..). You can then choose to do this incrementally (if your communication lines are slow).

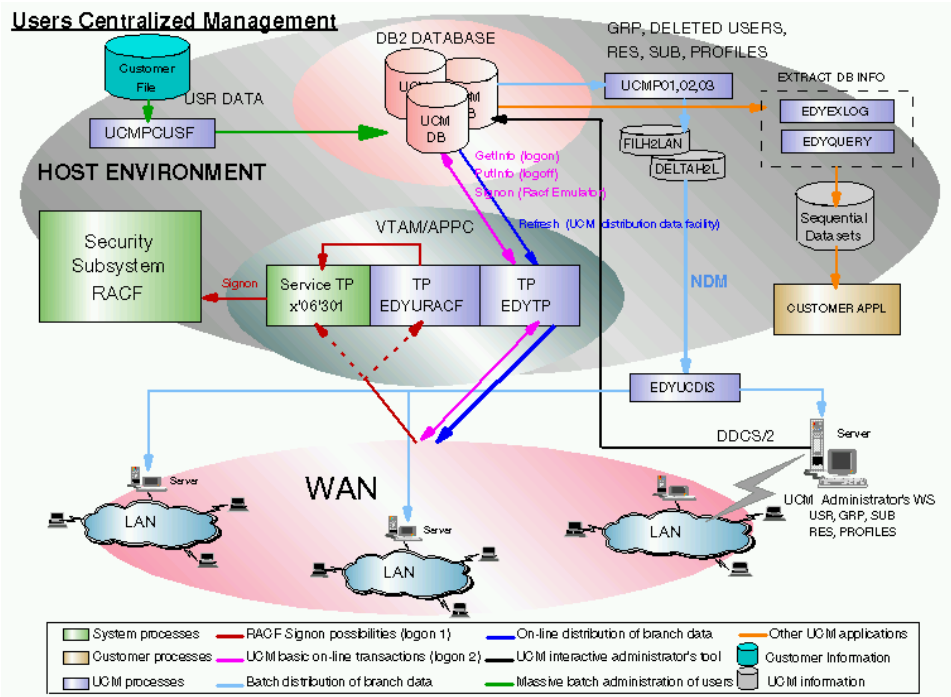
BATCH

You can gather through batch processes all branch information and distribute it through your own Software to the branches. Once this data is in the branches, it can be updated through the provided unattended procedures.

Through your own distribution software

For if you want UCM to centralize only the authentication process, but not to manage user's security policies. In this case you must distribute the security profiles through your own methods, and use the provided SecureEntry administration tools to make these active in the branches.

In the following picture you can see a full *Context Diagram* of the different UCM pieces and processes. Also shown at a high level is the information flow between each component.



Before proceeding with the UCM installation, you must decide on what installation mode will be used in your corporation according to the possibilities explained above. This implies deciding about which will be your main authentication subsystem, and whether you require UCM to manage the Security policies repository (suggested) or not. You must also choose between the on-line (suggested) and batch branch data distribution methods.

You can install RACF authentication (logon 1 **and** (optionally) the UCM basic transactions), or UCM authentication (logon 2 **and** (mandatory) the UCM basic transactions). Refer to UCM transaction programs for further information about those.

Once you have installed and customized the UCM Software, you will need to define the Groups and Resources through the UCM administrator's workstation. Then, you will define the SecureEntry users in UCM Database. You can add all your SecureEntry users through the UCMPCUSF utility. UCMPCUSF allows you to massively administer all your users, by managing the users additions, deletions and updates. You can also define your users with the standard SecureEntry administration tools (interactively or in batch mode), by running them against the UCM repository.

Once you have done all the UCM branch definitions you can download them to your corporate branches either:

In *In batch* mode

You can gather all your branch data information through UCMP01, UCMP02 and UCMP03 running in the Host environment and update it in your branches through EDYUCDIS running in the LAN environment.

These UCM processes gather the branch information and store it in a file (dataset under MVS). You must distribute this dataset to all your servers and then update the contained definitions through a SecureEntry LAN utility (EDYUCDIS).

In *On-line* mode

If this function (Dynamic refresh feature) is active, then at the specified time UCM will download the defined branch information to all your corporate branch servers through the EDYTP transaction program.

UCM provides the EDYEXLOG and EDYQUERY applications that allow you to extract information from the UCM Database in order to be processed by your own applications.

You can customize and run the UCM function in a MVS/ESA or OS/2 environment.

UCM under MVS/ESA has been designed for big corporations with a large number of users and branches. This corporations, usually demand a big amount of process capacity and high availability.

UCM under OS/2 has been designed for those corporations with a very limited number of users and branches. It is also the best solution for those corporations that can not afford or justify a 390 mainframe.

Note that before using the UCM OS/2 host code in a production environment, you must unlock the evaluation code, which is supplied together with SecureEntry/2 code, as explained under Installing UCM (OS/2 environment) so that no expiration date applies.

You can now proceed to the Installing UCM chapter, Where the different tasks involved in setting up the UCM code are detailed.

Software requirements

This section lists the software required for UCM installation and operation. There are different software requirements for the host (MVS or OS/2), the UCM administrator's workstation, and the branch workstations.

If you are planning to customize and install the UCM host code under OS/2, you don't need to read the Host requirements.

Host

UCM administrator's workstation/UCM Host workstation (UCM under OS/2)

Branches

Host (UCM under MVS)

MVS/ESA Version 4 Release 3 (5695-047, 5695-048)

RACF/MVS Version 1 Release 9.2 (5740-XXH) or other security subsystem at the same level. Note that you can instead optionally install UCM with RACF emulator feature. Refer to the SecureEntry administrator's guide for further information about it.

ACF/VTAM Version 3 Release 4 (5685-085)

ACF/NCP Version 4 Release 3 (5668-854)

ACF/SSP Version 4 Release 6 (5665-338)

C/370 or PL/I run-time libraries

DB2 MVS Version 2 Release 3 (5665-DB2)

UCM administrator's workstation / UCM Host workstation (UCM under OS/2)

OS/2 Warp V3.0 with FixPack XR_W017, or later, and (optionally) OS/2 Security Enabling Services

Communications Manager/2

DB2 and DDCS/2 Single User (DDCS/2 only with MVS/ESA customizations)

SecureEntry Release 3.0

Branches

OS/2 Warp V3.0 with FixPack XR_W017, or later, and (optionally) OS/2 Security Enabling Services

IBM LAN Server 4.0

Communications Manager/2 in the server domain controller

SecureEntry Release 3.0

Installing UCM

This chapter describes the tasks involved in installing UCM in the host (MVS/ESA and OS/2).

You will find the customization steps under each environment, as follows:

Installing UCM (MVS/ESA environment)

Installing UCM (OS/2 environment)

Installing UCM (MVS/ESA environment)

This section describes the tasks you must perform to install and customize UCM under MVS/ESA environment.

Customizing APPC/MVS

Creating the UCM database

Initializing the UCM database

Installing the UCM programs

Planning the distribution

Customizing the distributed data facility (DDF) feature of DB2 for MVS/ESA.

It also describes additional tasks you have to perform at the branch server and the UCM workstation when using UCM:

Configuring Communications Manager/2

Customizing DB2 DDCS/2 in the UCM workstation.

Binding the UCM API

Binding the Branch Query Application

Binding the EDYRVUCM recovery tool

If you install UCM using the provided RACF emulator, you will find a section describing the steps you must follow:

Installing RACF emulator

If you are installing RACF users authentication and you require data scrambling through the communications channel you must install and customize the EDYURACF TP:

Installing EDYURACF

In case you are migrating from an earlier UCM version, please follow:

Migrating from earlier UCM versions

***** NOTE *****

For a description of the SecureEntry related procedures to follow within SecureEntry itself during SecureEntry installation to enable UCM management, please refer to the 'Installing SecureEntry' chapter in the 'SecureEntry administrator's guide'.

Customizing APPC/MVS

APPC/MVS is a MVS/ESA standard component. To customize APPC/MVS, you must perform the following steps:

Customize APPC and ASCH in the parmlib

Set up the automatic startup of APPC and its Scheduler

Create two VSAM files needed by APPC

Define a major node and a specific logmode.

This steps are detailed in the following paragraphs:

1. Customizing the PARMLIB

   The information required for APPC and its scheduler, ASCH, is defined in the APPCPMxx and ASCHPMxx members of the PARMLIB. These members can be customized with the distributed data set members **UCM.Vxx.JCL(APPCPM00)** and **UCM.Vxx.JCL(ASCHPM00)**.

2. Setting up the automatic startup of APPC and its Scheduler

   You can include the commands to start APPC and ASCH using the SYS1.PARMLIB(COMMAND00) or the automatic startup procedure you normally use at IPL.

   The commands to start the APPC/MVS tasks are:

   ```
   S APPC,SUB=MSTR,APPC=xx
   ```

   ```
   S ASCH,SUB=MSTR,ASCH=xx
   ```

   Where xx corresponds to the suffix xx of the APPCPMxx and ASCHPMxx members of the PARMLIB.

3. Creating the necessary VSAM files

   APPC/MVS keeps the TP (transaction programs) information and the SIDE INFORMATION (an alias to define LUNAME,MODENAME and TPNAME) in two VSAM files. The following sections show how to create these VSAM files:

   IDCAMS to create the SIDE INFORMATION

   Use the distributed data set member **UCM.Vxx.JCL(ATBSIVSM)** as a working sample to create the side information VSAM file.

   IDCAMS to create the VSAM file that keeps the TP information

   Use the distributed data set member **UCM.Vxx.JCL(ATBTPVSM)** as a working sample to create the TP information VSAM file.

4. Defining a major node and a specific logmode

   These definitions are provided as a sample; Only required is the logmode name (APPCHOST) and the support of LU6.2 parallel sessions.

   VTAM major node

   Define this major node in SYS1.VTAMLIST or in the appropriate library.

```
*/* SYS1.VTAMLST APPL DEFINITIONS FOR APPC
*/*
APPLAPPC VBUILD TYPE=APPL
*
```

```
APPCLU1  APPL  ACBNAME=APPCLU1,
               APPC=YES,
               AUTOSES=0,
               DDRAINL=NALLOW,
               DLOGMOD=APPCHOST,
               DMINWNL=0,
               DMINWNR=3,
               DRESPL=NALLOW,
               DSESLIM=3,
               EAS=90,
               LMDENT=19,
               MODETAB=TABPCLU,
               PARSESS=YES,
               SECACPT=CONV,
               SRBEXIT=YES,
               VPACING=1
```

Logmode for OS/2 workstations and APPC

```
  LU62P    MODEENT LOGMODE=APPCHOST,
                   FMPROF=X'13',
                   TSPROF=X'07',
                   PRIPROT=X'B0',
                   SECPROT=X'B0',
                   COMPROT=X'50B1',
                   SSNDPAC=X'00',
                   SRCVPAC=X'00',
                   RUSIZES=X'8585',
                   TYPE=0,
                   PSNDPAC=X'00',
                   PSERVIC=X'060200000000000000002F00'
```

Creating the UCM database

Distributed data set member **UCM.Vxx.DB2(CREAUCM)** has the SQL sentences required to create all UCM V4Rx database tables.

Distributed data set member **UCM.Vxx.DB2(CREASYNR)** has the SQL sentences required to create all UCM V4Rx database synonyms to resolve dynamic SQL requests from UCM the workstations. It is necessary if the workstation USERID does not match the ID used when the database tables were created.

From DB2V4, and as alternative to synonyms, you can code the BIND parameter, **DYNAMICRULES(BIND)**. This way, the qualifier used to resolve the dynamic SQL sentences will be the same as the qualifier specified in the BIND process.

Migrating from earlier versions only: Users migrating from previous UCM versions will use instead data set member **UCM.Vxx.DB2(UP40TUCM)** and **UCM.Vxx.DB2(UP40SUCM)**, to create the two new V4Rx tables, indexes and its synonyms.

You must bear in mind, however, that the size for this database has been decided thinking of a test environment. Whenever this database is used for production, it must be reseized with regard to the estimated number of users.

You can run the SQL sentences through SPUFI, using JCL or from the UCM workstation using DDCS/2.

Initializing the UCM database

Distributed data set member **UCM.Vxx.DB2(UCMSETUP)** contains SQL sentences required to initialize the T_KEYWORD UCM database table with all UCM V4Rx default keyword values.

Migrating from earlier versions only: Users migrating from previous UCM versions will use instead data set member **UCM.Vxx.DB2(UP40KUCM)**, to initialize existing T_KEYWORD UCM database table with the new keywords default values.

Installing the UCM programs

For UCM processes, both on-line and batch, to run properly, you must install the UCM programs and APIs in MVS/ESA. The datasets that contain executable files and the UCM binds must follow the customer's nomenclature, making the necessary changes to the STEPLIBs of the JCLs that require them.

To install the UCM programs, you must perform the following tasks:

Define the necessary datasets for UCM (see the following JCL)

```
//ALLOCDTS JOB (),MSGLEVEL=(1,1),CLASS=W,MSGCLASS=X
//*********************************************
//*                                           *
//*  JCL to allocate the necessary datasets   *
//*  for UCM                                  *
//*                                           *
//*********************************************
//STEP1    EXEC PGM=IEFBR14
//LOADDD  DD DSN=UCM.Vxx.LOADLIB,
//           DISP=(NEW,CATLG),
//           UNIT=SYSDA,VOL=SER=RM2005,
//           SPACE=(6144,(640,320,10),,CONTIG,ROUND),
//           DCB=(RECFM=U,BLKSIZE=6144)
//JCLDD   DD DSN=UCM.Vxx.JCL,
//           DISP=(NEW,CATLG),
//           UNIT=SYSDA,VOL=SER=RM2005,
//           SPACE=(6160,(25,10,5),,CONTIG,ROUND),
//           DCB=(RECFM=FB,LRECL=80,BLKSIZE=6160)
//BNDDD   DD DSN=UCM.Vxx.BND,
//           DISP=(NEW,CATLG),
//           UNIT=SYSDA,VOL=SER=RM2005,
//           SPACE=(6160,(25,10,5),,CONTIG,ROUND),
//           DCB=(RECFM=FB,LRECL=80,BLKSIZE=6160)
//OBJDD   DD DSN=UCM.Vxx.OBJ,
//           DISP=(NEW,CATLG),
//           UNIT=SYSDA,VOL=SER=RM2005,
//           SPACE=(6160,(25,10,5),,CONTIG,ROUND),
//           DCB=(RECFM=FB,LRECL=80,BLKSIZE=6160)
//DB2DD   DD DSN=UCM.Vxx.DB2,
//           DISP=(NEW,CATLG),
//           UNIT=SYSDA,VOL=SER=RM2005,
//           SPACE=(6160,(25,10,5),,CONTIG,ROUND),
//           DCB=(RECFM=FB,LRECL=80,BLKSIZE=6160)
//*
//* ************************************************************
//* Sample data set of a parameter's EDYEXLOG file.            *
//* ************************************************************
//PARAMS    DD DSN=UCM.Vxx.EDYEXLOG.PARAMS,
//           DISP=(NEW,CATLG),
//           UNIT=SYSDA,VOL=SER=RM2005,
//           SPACE=(CYL,(2,1)),
//           DCB=(RECFM=FB,LRECL=80,BLKSIZE=800,DSORG=PS)
//*
//* ************************************************************
//* This data set is optional. It hosts one data set member with    *
//*   customer data translation table:                              *
//*      ASC2EBC  (ASCII to EBCDIC translation table).              *
//* If omitted, default data translation table will apply.         *
//* ************************************************************
//TABLESDD  DD DSN=UCM.Vxx.TABLES,
//           DISP=(NEW,CATLG),
```

```
//           UNIT=SYSDA,VOL=SER=RM2005,
//           SPACE=(CYL,(2,1)),
//           DCB=(RECFM=FB,LRECL=256,BLKSIZE=2560)
/*
```

Copy the UCM datasets from the installation cartridge (see the following JCL)

```
//TAPEDISK JOB (),MSGLEVEL=(1,1),CLASS=W,MSGCLASS=X
//*********************************************
//*                                           *
//*  JCL to copy the UCM datasets from the    *
//*  installation cartridge                   *
//*                                           *
//*********************************************
//STEP1    EXEC PGM=IEBCOPY
//SYSUT3  DD   UNIT=SYSDA,SPACE=(CYL,(5,5))
//SYSUT4  DD   UNIT=SYSDA,SPACE=(CYL,(5,5))
//LOADDS  DD DSN=UCM.Vxx.LOADLIB,DISP=SHR
//LOADTP  DD DSN=UCM.Vxx.LOADLIB,
//           DISP=(,PASS),LABEL=(1,SL),
//           UNIT=TAP01,VOL=SER=UCMTAP,
//           DCB=(RECFM=U,BLKSIZE=6144)
//JCLDS   DD DSN=UCM.Vxx.JCL,DISP=SHR
//JCLTP   DD DSN=UCMTAPE.UCM.JCL,
//           DISP=(,PASS),LABEL=(2,SL),
//           UNIT=TAP01,VOL=SER=UCMTAP,
//           DCB=(RECFM=FB,LRECL=80,BLKSIZE=6160)
//BNDDS   DD DSN=UCM.Vxx.BND,DISP=SHR
//BNDTP   DD DSN=UCM.Vxx.BND,
//           DISP=(,PASS),LABEL=(3,SL),
//           UNIT=TAP01,VOL=SER=UCMTAP,
//           DCB=(RECFM=FB,LRECL=80,BLKSIZE=6160)
//OBJDS   DD DSN=UCM.Vxx.OBJ,DISP=SHR
//OBJTP   DD DSN=UCM.Vxx.OBJ,
//           DISP=(,PASS),LABEL=(4,SL),
//           UNIT=TAP01,VOL=SER=UCMTAP,
//           DCB=(RECFM=FB,LRECL=80,BLKSIZE=6160)
//DB2DS   DD DSN=UCM.Vxx.DB2,DISP=SHR
//DB2TP   DD DSN=UCM.Vxx.DB2,
//           DISP=(,PASS),LABEL=(5,SL),
//           UNIT=TAP01,VOL=SER=UCMTAP,
//           DCB=(RECFM=FB,LRECL=80,BLKSIZE=6160)
//PARDS   DD DSN=UCM.Vxx.EDYEXLOG.PARAMS,DISP=SHR
//PARTP   DD DSN=UCM.Vxx.EDYEXLOG.PARAMS,
//           DISP=(,PASS),LABEL=(6,SL),
//           UNIT=TAP02,VOL=SER=UCMTAP,
//           DCB=(RECFM=FB,LRECL=80,BLKSIZE=800)
//TABDS   DD DSN=UCM.Vxx.TABLES,DISP=SHR
//TABTP   DD DSN=UCM.Vxx.TABLES,
//           DISP=(NEW,PASS),LABEL=(7,SL),
//           UNIT=TAP02,VOL=SER=UCMTAP,
//           DCB=(RECFM=FB,LRECL=256,BLKSIZE=2560)
/*
//SYSPRINT  DD  SYSOUT=*
       COPY INDD=LOADTP,OUTDD=LOADDS
       COPY INDD=JCLTP,OUTDD=JCLDS
       COPY INDD=BNDTP,OUTDD=BNDDS
       COPY INDD=OBJTP,OUTDD=OBJDS
       COPY INDD=DB2TP,OUTDD=DB2DS
       COPY INDD=PARTP,OUTDD=PARDS
       COPY INDD=TABTP,OUTDD=TABDS
/*
```

Allocating additional datasets

The data set member **UCM.Vxx.JCL(ALLOCDT2)** is distributed to allocate the additional datasets:

UCMLOG

UCMLOG is used to keep a log in the MVS environment. It is used by the UCMPCUSF process.

FILH2LAN

FILH2LAN is allocated to contain the information that is to be distributed to the branches. This DD is referenced in the UCMP03 process.

DELTAH2L

DELTAH2L is allocated to contain the information that is to be distributed to the branches. This dataset can be referenced in the UCMP01 and UCMP03 processes instead of FILH2LAN, if you want to maintain two separate versions (main and delta information).

SEQLOG

SEQLOG is allocated to receive the output of EDYEXLOG.

UCMQUERY

UCMQUERY is allocated to receive the output of EDYQUERY.

Perform a bind of the UCM programs.

The data set member **UCM.Vxx.JCL(BIND)** is a working sample of a JCL to perform the BIND, using members from distributed library UCM.Vxx.BIND:

UCMPCUSF

UCMP01

UCMP02

UCMP03

PUTGETUS

UCMREFSH

EDYEXLOG

EDYQUERY

Define the EDYTP TP in the APPC corresponding VSAM using the ATBSDFMU program with the TPADD parameter. As a sample, you can see the distributed member **UCM.Vxx.JCL(TPPROF)**. If you use this member you must change it so that it matches the customer's nomenclature.

Use distributed dataset members **UCM.Vxx.JCL(ATBTPVSM)** and **UCM.Vxx.JCL(ATBSIVSM)** as working samples to define the VSAM DATASETs to store APPC information. This is useful if you are installing APPC for the first time.

Copy and adapt to the customer's nomenclature the batch processes JCLs distributed on data set **UCM.Vxx.JCL**:

UCMPCUSF

UCMP01

UCMP02

UCMP03

EDYEXLOG

EDYQUERY

Planning the distribution

To distribute the information about resources, groups, and links among groups and resources in the various subsystems (SecureEntry, LAN Server), you can use the UCM V4Rx distribution on-line system.

Alternatively, you can collect branch changes with the UCM batch processes and distribute those with whatever distribution system you currently use for your installation.

UCM V4Rx on-line process.

UCM V4Rx has administration tools to activate the on-line branch refresh feature (Refresh Branch On-line). Trough this tools, you can select from a list of several options when UCM changes will be updated into the branches.

EDYSRV.EXE, a process running in LAN environment, will update this changes at every corporate branch.

For further information about this process you can see SecureEntry administrator's guide

UCM batch processes.

UCM provides the processes UCMP01, UCMP02, and UCMP03 to collect information from the database. These processes are described in full detail in Users Centralized Management (UCM) processes.

If you want to do batch update of the branches data (by not using the refresh branch on-line UCM feature), then you will have to design two distribution plans:

PLAN1

Run the UCMP03 process to get all the necessary information in order to create an object to initialize a new branch. Distribute this object to the new branch. Run the EDYUCDIS batch process at the branch servers using the distributed object as input parameter. It will initialize all the branch workstations.

PLAN2

Run the UCMP01 process to get all the modified information in order to create an object to update the branch information. Distribute this object to the already initialized branches. Run the EDYUCDIS batch process at the branch server using the distributed object as input parameter. It will update all the branch workstations.

Information about users is distributed from the host to the branches, or the other way, through the UCM communication channel. This distribution is always on-line at logon and logoff times.

Customizing the DDF feature of DB2 for MVS/ESA

Since the UCM workstation access to DB2 for MVS/ESA is based in the distributed relational database architecture (DRDA), the DDF feature of DB2 for MVS/ESA must be configured to enable this access.

If DDF is not configured, refer to DBS for MVS/ESA, Installation Guide for information on configuring it.

Configuring Communications Manager/2

The UCM installation requires customization of the communications in order for the DRDA product to be able to access the UCM repository from within the UCM administrator's workstation and to enable the communication channel used to authenticate users in a centralized way.

You must upgrade your CM/2 configuration in the server workstation of every branch and in the UCM administrator's workstation as explained below:

Customizing CM/2 in the UCM administrator's workstation

Customizing CM/2 in the Server workstations

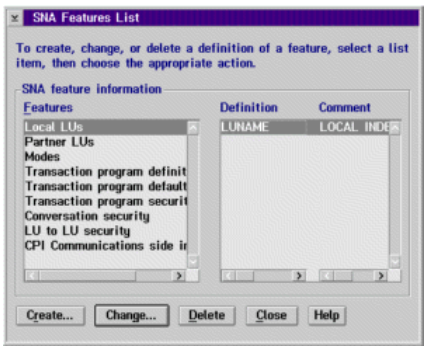
Customizing CM/2 in the UCM administrator's workstation

This section consists of a step-by-step list of the definitions you have to specify to upgrade your CM/2 configuration for the UCM administrator workstation. This definitions are a superset of the required definitions for a regular branch server workstation:

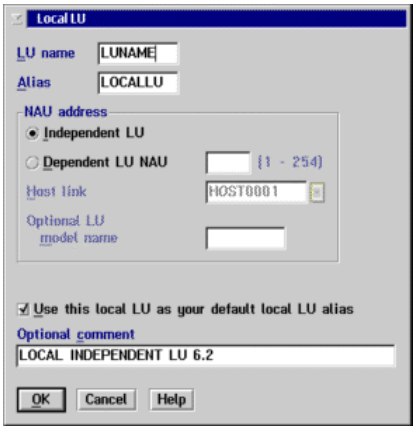
1. In the Communications Manager Configuration List panel, select the **SNA features** option.

2. Create the local independent LU (6.2) that will establish the APPC connections. This LU can be an independent LU defined in the host with a LOCAL ADDRESS=0, or use the CP. The alias name can be LOCALLU.

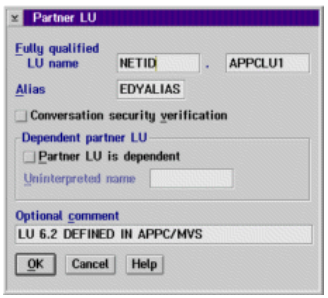
   If you create a local independent LU 6.2 it should be configured as the default LU 6.2. If you need to use a not-default one, you must make use then of the SecureEntry/2 environment variable **SGM_FORCE_LUALIAS**.



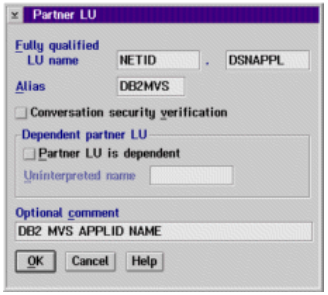
3. In this sample, the LU LUNAME is defined as the default independent local LU with alias LOCALLU. The LUNAME (if CP is not used) should be provided by your SNA network administrator. The LU alias name can be anyone you choose.



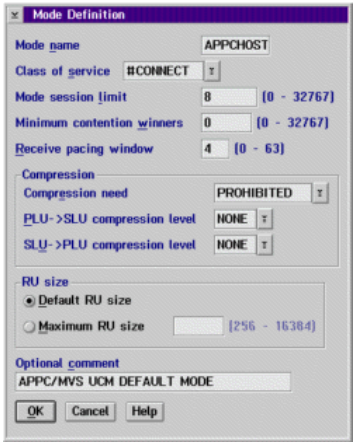
4. Define the partner LU defined in the host as the LU 6.2 with support for parallel sessions, APPC=YES, and logmode APPCHOST. This is the door to the APPC/MVS. Type in its fully qualified LU name and the alias, which must be EDYALIAS. In the following sample the networkId is NETID and the partner LU is APPCLU1.



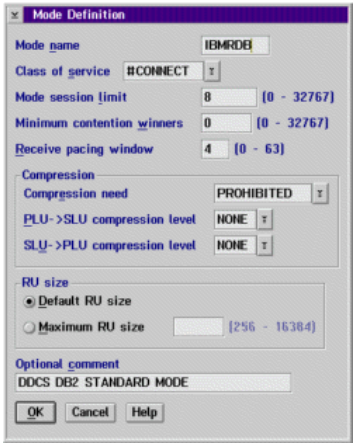
5. Define this partner LU for the UCM workstation. This is the partner DB2 definition to support DRDA, thus enabling the SecureEntry Administration Application access to the UCM central database. Define the partner LU defined in the host as the DB2 subsystem APPL and type an alias to be used in the CPI definitions.



6. Add the Mode Definition with the standard parameters. The only specific requirement is the Mode name, which must be APPCHOST.

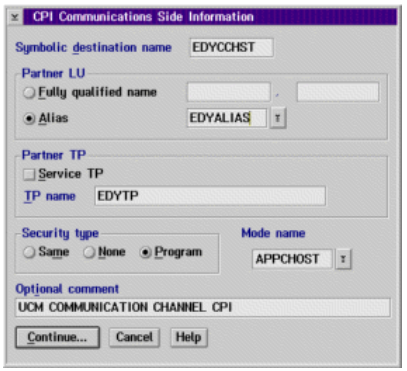


7. Define this Mode Definition for the UCM workstation. The only required parameter is the Mode name, since DB2 in MVS has the default logmode IBMRDB. If you want to change this name, call your host DB2 system administrator. The parameters specified in this mode are negotiated during session establishment.



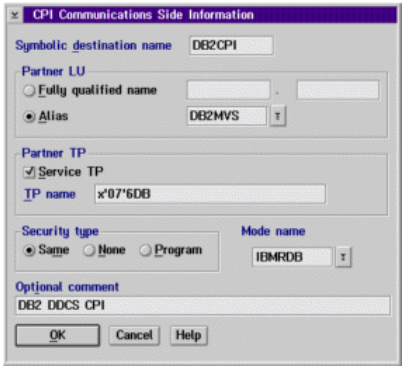
8. The CPI Communications Side Information includes the LU names and TP names of programs that you can access through the CPI Communications subsystem. You must define the CPI EDYCCHST. It will be used as the UCM communication channel between the servers in the branches and the part of the UCM application that resides in the host.

   The name of the UCM TP in the host is EDYTP and the logmode used to establish the session is APPCHOST. The security type is program, which means that a UserId and a password will be sent when the communications session is established.



9. Press the **Continue..** button and specify the userID defined in RACF, with a password that does not expire, and has the necessary grants to access the UCM database. This userID authority will be used by the UCM application to get and put information from the branches into the UCM database.

10. You must define the CPI DB2CPI. This definition correlates with the DB2 Database Director definitions that link the DB2 alias UCM with the DB2 node location. The service TP that resides in the MVS DB2 to allow the remote connection name is x'07'6DB. The security type is the same. The logmode is IBMRDB.



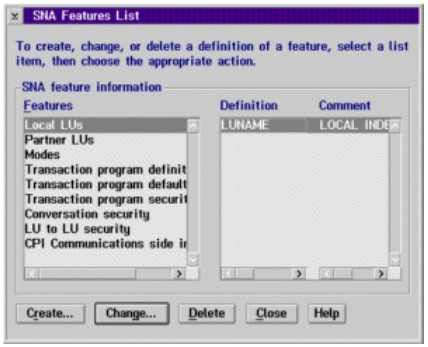
Customizing CM/2 in the server workstation

This section consists of a step-by-step list of the definitions you have to specify to upgrade your CM/2 configuration for the Server workstations:

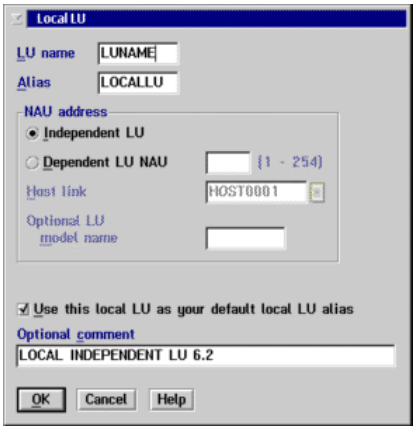
1. In the Communications Manager Configuration List panel, select the **SNA features** option.

2. Create the local independent LU (6.2) that will establish the APPC connections. This LU can be an independent LU defined in the host with a LOCAL ADDRESS=0, or use the CP. The alias name can be LOCALLU.

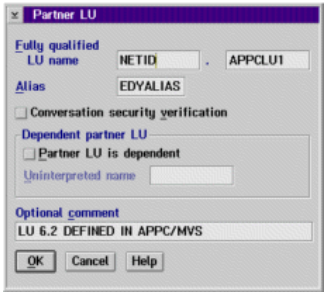
   If you create a local independent LU 6.2 it should be configured as the default LU 6.2. If you need to use a not-default one, you must make use then of the SecureEntry/2 environment variable **SGM_FORCE_LUALIAS**.



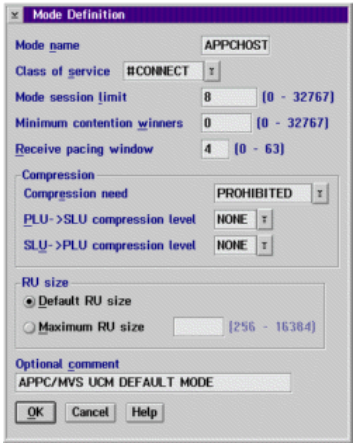
3. In this sample, the LU LUNAME is defined as independent LU by default with the alias LOCALLU. The LUNAME should be provided by your SNA network administrator. The LU alias name can be defined as you wish.



4. Define the partner LU defined in the host as the LU 6.2 with support for parallel sessions, APPC=YES, and logmode APPCHOST. This is the door to the APPC/MVS. Type in its fully qualified LU name and the alias, which must be EDYALIAS. In the following sample the networkId is NETID and the partner LU is APPCLU1.

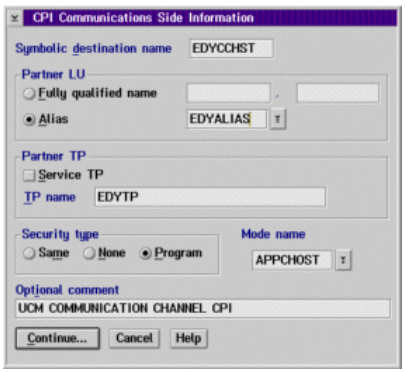


5. Add the Mode Definition with the standard parameters. The only specific requirement is the Mode name, which must be APPCHOST.



6. The CPI Communications Side Information includes the LU names and TP names of programs that you can access through the CPI Communications subsystem. You must define the CPI EDYCCHST. It will be used as the UCM communication channel between the servers in the branches and the part of the UCM application that resides in the host.

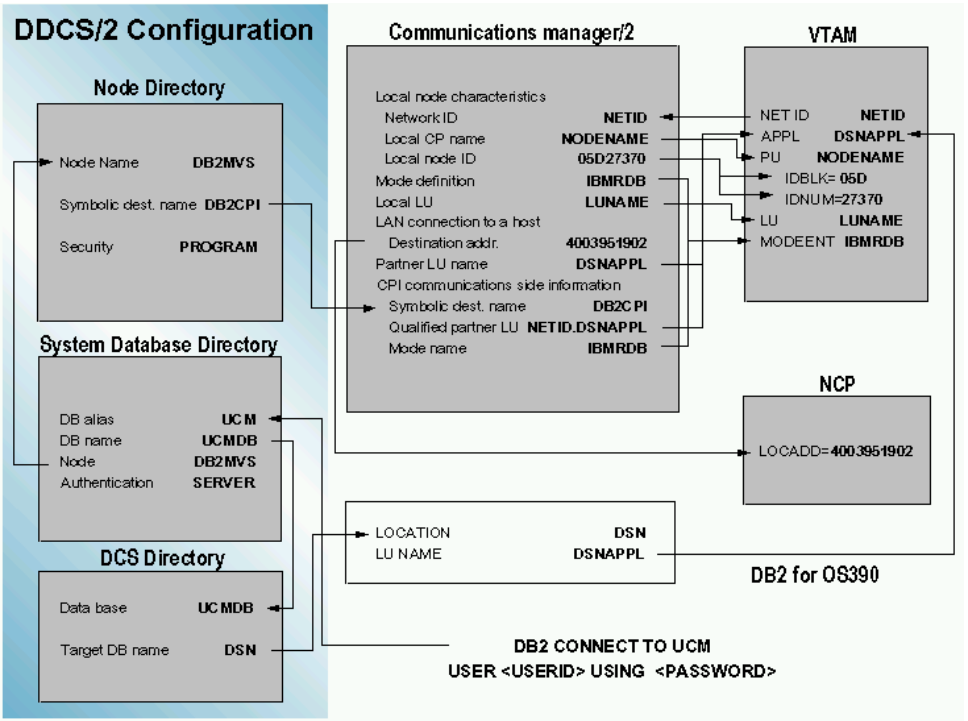
   The name of the UCM TP in the host is EDYTP and the logmode used to establish the session is APPCHOST. The security type is program, which means that an UserId and a password will be sent when communications session is established.



7. Press the **Continue..** button and specify the userID defined in RACF, whose password does not expire, and has the necessary grants to access the UCM database. This userID authority will be used by the UCM application to get and put information from the branches to the UCM database.

Customizing DB2 DDCS/2 in the UCM workstation

To prepare the UCM workstation, you must add the information shown in the following picture to DB2 DDCS/2 configuration to connect the remote UCM database with the SecureEntry administration application:



You can define the UCM Database access through the graphic interface or from an OS/2 window as explained underneath:

You must catalog the APPC node with the symbolic destination name defined in CM/2.

DB2 CATALOG APPC NODE **DB2MVS** REMOTE **DB2CPI** SECURITY PROGRAM/SAME/NONE

Catalog the database in the System Database Directory as follows:

DB2 CATALOG DB **DBUCM** AS UCM AT NODE **DB2MVS** AUTHENTICATION SERVER

Catalog in the DCS Directory the UCM database:

DB2 CATALOG DCS DB **UCMDB** AS **DSN**

Where DSN is the name of the DB2 MVS/ESA subsystem shared by DDF. Through Distribute Data Facility (DDF MVS/ESA) you can share your DB2 subsystem to another systems. You must define the parameter **LOCATION** and **LU NAME**. LOCATION is the name your DB2 will be known by other systems, and LU NAME is the independent local LU 6.2 through which DB2 will be accessed.

It is very important that you bind the *DDCS/2 applications and utilities* as documented in **DDCS/2 User's Guide**.

Binding the UCM API

To administer the UCM database from the UCM workstation, you can use the UCM API. Because embedded SQL has been used to develop this API, it must be bound to each host-based database with which the API will interact.

Access plans are contained in the *bind files*, which are created during precompilation. Binding just means the processing of these files by a DRDA server.

To bind the UCM API, you should make sure that you are assigned sufficient authority in the MVS database management system:

```
SYSADM   OR
SYSCTRL  OR
BINDADD and CREATE IN COLLECTION NULLID and/or BIND
```

Then, issue the following statement:

```
DB2START
DB2 CONNECT TO dbalias USER userid USING password
DB2 BIND @EDYUCM.LST QUALIFIER ucmsqid
DB2 CONNECT RESET
DB2STOP
```

Where *dbalias* is the host database alias defined in DDCS/2 and *userid* is a user ID with the appropriate grants to bind the programs. *ucmsqid* corresponds to the prefix defined on the UCM database.

Note that you can do the above process by using the EDYUCBND command, located within the SecureEntryPath, install directory. Invoke it without parameters to read about the invocation syntax.

Processing the bind files in separate packages provides the necessary granularity to grant different execute privileges at a user or group level.

If you sign-on using a userid with SYSADM privilege, you can run any UCM API package. Regular users should be granted the execute privilege on the API packages, depending on the tasks they are to be authorized to perform.

These are the provided administration plans:

Basic access

The user can only view the database information.

User access

The user can view the database information and change the information related to the user objects.

Group and resource access

The user can view the database information and change the information related to the defined groups or resources.

Full access

The user can perform any task within the database.

To simplify the grant task, a command line utility is provided to perform the appropriate grant execute on package statements to a user or group:

```
EDYUCM GRANT /G:id /F:filename [/D:database] [/P:prefix] [/CONTINUE]
```

id

User ID or group ID to which the privilege is to be granted.

filename

Name of the file that contains the list of packages for which the privilege is to be granted.

database

Database alias. The default is **UCM**.

prefix

Package prefix name. If the parameter is omitted, the package prefix must be specified in the filename, in the form prefix.package_name.

CONTINUE

Specifies that the command is to be executed for every entry in the file, even if errors occur. If the parameter is omitted, the command will only complete if no errors are found during processing.

These are the provided files for each type of access:

EDYBASIC.LST

Basic access

EDYUSER.LST

User access

EDYGROUP.LST

Group and resource access

EDYALL.LST

Full access

To revoke the execute privilege from a user or group, the following utility is provided:

```
EDYUCM REVOKE /G:id /F:filename [/D:database] [/P:prefix] [/CONTINUE]
```

id

User ID or group ID from which the privilege is to be revoked.

filename

Name of the file that contains the list of packages on which the privilege is to be revoked.

database

Database alias. The default is **UCM**.

prefix

Package prefix name. If the parameter is omitted, the package prefix must be specified in the filename, in the form prefix.package_name.

CONTINUE

Specifies that the command be executed for every entry in the file, even if errors occur. If the parameter is omitted, the command will only complete if no errors are found during processing.

The following sample command grants the minimum authorization level to the USER0112 user.

```
edyucm grant /G:USER0112 /F:EDYBASIC.LST
```

This other sample grants full authorization to the ADMINS group on the UCM database; the prefix of the UCM database tables (UCMPROD) is explicitly specified.

```
edyucm grant /G:ADMINS /F:EDYALL.LST /P:UCMPROD
```

Binding the Branch Query Application

To query the branch level information of your branches you can use the EDYQRYBR process in the LAN Environment. Because embedded SQL has been used to develop this Application, it must be bound to each host-based database with which the process will interact.

The access plan is contained in the *bind file*, which is created during precompilation. Binding just means the processing of this file by a DRDA server.

To bind the EDYQRYBR Application, you should ensure that you are assigned sufficient authority in the MVS database management system:

```
SYSADM   OR
SYSCTRL  OR
BINDADD and CREATE IN COLLECTION NULLID and/or BIND
```

Then, issue the following statement:

```
DB2START
DB2 CONNECT TO dbalias USER userid USING password
DB2 BIND @EDYQRYBR.LST QUALIFIER ucmsqid
DB2 CONNECT RESET
DB2STOP
```

Where *dbalias* is the host database alias defined in DDCS/2 and *userid* is a user ID with the appropriate rights to bind the programs. *ucmsqid* corresponds to the prefix defined on the UCM database.

Note that you can do the above process by using the EDYUCBND command, located within the SecureEntryPath, install directory. Invoke it without parameters to see the syntax.

The **EDYQRYBR.LST** file located within SecureEntryPath, install directory can be used to bind the process.

As we have explained in the previous chapter, you can use the EDYUCM command line utility to perform the appropriate grant task. To do this, you can use the **EDYQUERY.LST** file as provided and located within the SecureEntry path, install directory.

Binding the EDYRVUCM recovery tool

This tool's purpose is to correct and compress the change tables of the UCM Database when an error is detected in the Refresh branch on-line process. Because embedded SQL has been used to develop this Application, it must be bound to each host-based database with which the process will interact.

The access plan is contained in the *bind file*, which is created during precompilation. Binding just means the processing of this file by a DRDA server.

To bind the EDYRVUCM Application, you should ensure that you are assigned sufficient authority in the MVS database management system:

```
SYSADM   OR
SYSCTRL  OR
BINDADD and CREATE IN COLLECTION NULLID and/or BIND
```

Then, issue the following statement:

```
DB2START
DB2 CONNECT TO dbalias USER userid USING password
DB2 BIND @EDYRVUCM.LST QUALIFIER ucmsqid
DB2 CONNECT RESET
DB2STOP
```

Where *dbalias* is the host database alias defined in DDCS/2 and *userid* is a user ID with the appropriate grants to bind the programs. *ucmsqid* corresponds to the prefix defined on the UCM database.

Note that you can do the above process by using the EDYUCBND command, located within the SecureEntryPath, install directory. Invoke it without parameters to read about the syntax.

The **EDYRVUCM.LST** file, located within the SecureEntryPath, install directory, can be used to bind the process.

As it has been explained in the previous chapter, you can use the EDYUCM command line utility to perform the appropriate grant task. To do this, you can use the **EDYRCVRY.LST** file as provided, which is located within the SecureEntry path, install directory.

Installing RACF emulator

If you plan to install the RACF emulator feature you can do it through the SecureEntry installation tool. Refer to the SecureEntry administrator's guide for further information on how to do it.

When you choose this installation option, you must specify the **UCMINSTT DD** for the following runnable processes:

1. UCMPCUSF
2. EDYTP

This **DD** must point to a member data set which keeps information related to the corporation name, the allowable character codes and maximum and minimum length for passwords validation as needed by the RACF SecureEntry emulator. There is one description key for each value that you can specify.

In this data set you can specify the following parameters (keys):

INSTITUTION = Corporation name. Must be the same as used during SecureEntry installation in the branches.

MINPASSWORDLEN = Minimum password length allowed. Must be the same as used during SecureEntry installation.

MAXPASSWORDLEN = Maximum password length allowed. Must be the same as used during SecureEntry installation.

AVAILABLECODES = Character set allowed for password validation. Must be the same as used during SecureEntry installation.

UCMSUBSRACFKEY = Scrambling key for secure RACF authentication through EDYURACF.

Use distributed data set member **UCM.Vxx.TABLES(INSTITUT)** as a working sample to create the password information file.

If you are already a UCM user, then you will see the performance of the UCM logon transactions increased. With the RACF emulator, the logon process will normally execute only one transaction instead of the two Host-transactions executed when you use the RACF feature. SecureEntry will use the authenticate/validate user/password transaction to know if there are any changes for the user logging in. If there are no changes, then the transaction to download the user definition changes will not be executed.

Remember: It is not possible to migrate from/to RACF validation to the RACF SecureEntry emulator feature in an automated way. If you want to do it, please, contact the SecureEntry support team.

Installing EDYURACF

EDYURACF is a *transaction program* designed to receive user authentication transactions coming from the different branches, descramble the input data, and establish an APPC conversation against RACF to finally authenticate the users.

In order to enable this feature, you have to follow a list of configuration steps both in the branch servers as in the MVS/ESA host environment.

Branch servers (including the UCM administrator's workstation)

Do the following:

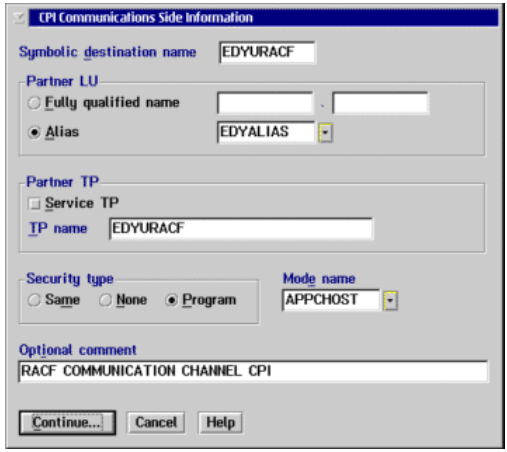
1. Whenever you install SecureEntry in a server machine with user's authentication option set to use RACF and with scrambled communications option checked, SecureEntry install program will copy the file **EDYURACF.DLL** located in *SecureEntryPath\INSTALL* directory to *SecureEntryPath\DLL*. By doing this, SecureEntry installation will implicitly enable the communications scramble feature.

   If you wish user authentication data to be scrambled with a master scrambling key value different than the INSTITUTION NAME, as specified during installation, you must add **UCMSUBSRACFKEY** key into file **EDYUCFPW.DSC** located in the *SecureEntryPath\INSTALL* directory.

2. You must add the following to your Communications Manager/2 configuration:

Define the CPI (Communications Side Information) EDYURACF. This definition will be used in order to establish a communication channel between the server workstations and the EDYURACF TP in MVS/ESA environment. The *partner LU* is **EDYALIAS**.

Follows a sample of EDYURACF CPI configuration:



The security type must be **Program**. and in the *CPI Communication program security* you have to define a UserID whose password does not expire in RACF as can be seen in the following figure:

If you require more information about EDYALIAS configuration or the LOGMODE APPCHOST you can read Communications Manager/2 configuration (steps **4** and **6**).

MVS/ESA environment

Under MVS/ESA environment, you must define a new TP named EDYURACF with the following requirements:

The EDYURACF TP must be stored in an **APF library**, because it establishes a communication session with a partner LU 6.2.

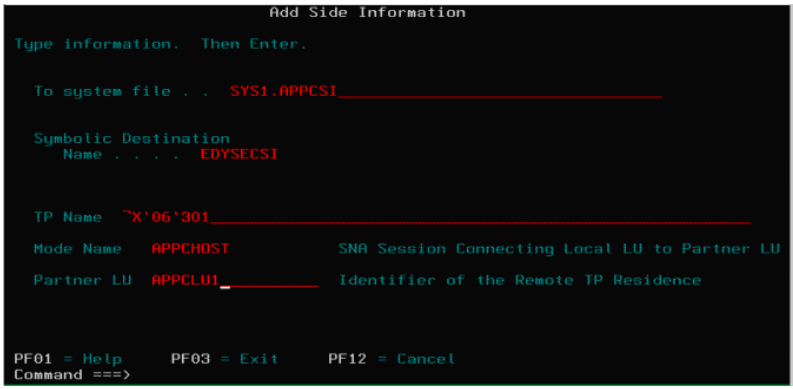
You must define the EDYURACF TP in the APPC corresponding VSAM using ATBSDFMU program with the TPADD parameter. As a sample, you can see the distributed member **UCM.Vxx.JCL(TPPROF)** Adapt it to your desired TP name, parameters and customer's nomenclature.

EDYURACF uses a master scrambling key to decipher the information received from the branches. This key can be provided in two different ways:

1. */I:decipher_key* load parameter.
2. **UCMINSTT** DD in the execution JCL of EDYURACF TP.

For further information about the scrambling key you can read EDYURACF transaction program .

You must customize the APPC CPI Side information **EDYSECSI**. This CPI must reference the partner local independent LU 6.2 existing in APPC/MVS (different LU 6.2 or not), and in the TP field you must customize the **SNA Service TP X'06'301**. The LOG MODE must be APPCHOST. You can see the above definitions in the following picture:



As an alternative to the ISPF/PDF panels you can use the ATBSDFMU program with **SIADD** parameter to add the EDYSECSI CPI definition.

Migrating from earlier UCM versions

You can upgrade to UCM V4Rx from an earlier version. Version 4 provides tools to update the group and resource definitions in the branches through EDYSRV and EDYTP processes (refresh branch on-line).

This a very useful function in order to dynamically update branch information that can be customized for your branches depending on their needs or requirements. This way, you will not need to use the provided UCMP01, UCMP02, UCMP03 and EDYUCDIS batch processes to distribute changes to LAN branches.

If you plan on upgrading to UCM version 4, you also will have to upgrade SecureEntry/2 to build level 190 or higher.

To update the UCM version in the Host environment you must follow this steps:

Create new UCM V4Rx tables.

Distributed data set member **UCM.Vxx.DB2(UP40TUCM)**, sets the SQL sentences to create the two new V4Rx tables and its indexes.

You can run the SQL sentences from SPUFI, using JCL or from the UCM workstation using DDCS/2.

Create new UCM V4Rx synonyms.

Distributed data set member **UCM.Vxx.DB2(UP40SUCM)** sets the SQL sentences to create the two new V4Rx synonyms.

You can run the SQL sentences from SPUFI, using JCL or from the UCM workstation using DDCS/2.

Load T_KEYWORD database.

Distributed data set member **UCM.Vxx.DB2(UP40KUCM)**, sets the SQL sentences to load existing T_KEYWORD UCM database table with the new UCM V4Rx keywords default values. sentences to create the two new V4Rx synonyms.

You can run the SQL sentences from SPUFI, using JCL or from the UCM workstation using DDCS/2.

Installing the UCM V4Rx distribution.

This is the same step as for a new installation. To have the UCM code updated to the latest level, we recommend you to install all distributed modules.

Follow the instructions in Installing the UCM programs.

Installing UCM (OS/2 environment)

If your UCM load requirements are low, or you do not have an MVS/ESA environment available, you can optionally install the UCM code under an OS/2 environment. Even if the function performed will be the same, you should be aware that you are installing different programs under a completely different environment. Because of that, the installation and configuration processes are completely different and explained underneath.

If you plan to install the UCM host code under OS/2, we suggest you to install and customize UCM in the same workstation that you will use afterwards to administer the UCM database through the SecureEntry provided administration tools.

Under this environment (OS/2), you will need to have the following:

OS/2 prerequisites

The SecureEntry/2 installation tool installs all necessary modules to customize UCM under OS/2 environment.

Step-by-step guide to installation:

UCM installation fast guide on OS/2

Detailed description of the involved processes:

Customizing Communications Manager/2

Creating the UCM Database

Binding the UCM programs

Additionally, the following customization tasks must be performed in the UCM host workstation before UCM becomes operative and accepts requests from your workstations:

Initializing the UCM Database

Planning data distribution

For last, you will find detailed descriptions and samples of the UCM processes, as well as samples of Communications Manager/2 configuration files for the **UCM workstation** and for the branch **Server workstation**.

UCM additional software

IMPORTANT NOTE

The UCM/2 code is distributed together with SecureEntry/2 only for evaluation purposes, and expires (becomes non-functional) in a 6 months timeframe after SecureEntry build creation date. It must be contracted separately before it can be set up in a production environment. At that time, our team will supply you with a password to unlock the evaluation license. Then, by running :

```
EDYTPPRD -P: password
```

from the SecureEntry path, *INSTALL* directory, your UCM/2 copy will become a production one, without expiration date.

OS/2 prerequisites

When installing UCM under OS/2 environment and since RACF is not available under this environment, you must install and customize SecureEntry to use the RACF emulator for users authentication. In this case UCM will obtain the 'Institution name' installation parameter (used to scramble the user's passwords into the UCM database) directly from the file SENTRY.CNF located in *\SGMSHELL\INSTALL* directory (as set up by the default SecureEntry/2 installation). EDYTP (the transaction program to serve connecting workstations) can also be customized to obtain the Institution name from a load parameter through CM/2 customization.

In addition, you will have to install the ASCII/EBCDIC translation tables located in the *\SGMSHELL\INSTALL* directory. These tables are used by UCM code to translate the incoming data to the host code set convention. Since under this environment the host code set is also ASCII because running under OS/2, both tables should contain all of the ASCII code points mapped not to perform any translation at all. You will find this 'passthrough' tables to be used with UCM OS/2 in the *\SGMSHELL\INSTALL\UCMOS2* directory. Please proceed to copy them to the *\SGMSHELL\INSTALL* directory. **This must be done in your UCM workstation and in all of your branch server workstations.**

UCM installation fast guide on OS/2

This chapter consist of a step-by-step guide of the definitions required for each environment:

UCM workstation

Under OS/2 environments, the UCM Database must be installed in the same workstation that will be used for administration tasks.

In order to install the UCM Database, and to activate the UCM centralized administration feature, you must follow the steps explained hereafter:

1. Customize Communications Manager/2.
2. Install SecureEntry/2 with UCM feature (without RACF, so that RACF emulator is used instead). Note that no machine reboot is required at this stage yet.
3. Edit the file **EDYSTART.CMD**,adding the following lines at the beginning:

```
edylkmsg "Starting DB2"
logon userID /P:password /L
```

```
    db2start

Where

userID: Must be an userID with enough grants to start DB2.
        We recommend that you define the same user as previously
        defined for EDYTP's communication channel authentication.
```

4. Reinitialize (reboot) the workstation and perform logon with a previously existing LAN administrator userid. This will be an emergency logon, since the UCM database has not been created yet.

5. Copy the translation files EDYA2E.DAT and EDYE2A.DAT located in the *\SGMSHELL\INSTALL\UCMOS2* path to the *\SGMSHELL\INSTALL* directory.

6. Use the CREAUCM.CMD utility to create the UCM Database.

7. Initialize the UCM Database by running the UCMSETUP.CMD utility.

8. Bind the UCM API and the UCM administration software by using the BINDUCM.CMD utility.

9. Run the **INSTSUB.EXE** program in order to initialize the UCM Database with the basic subsystem definitions. INSTSUB.EXE can be found in *\SGMSHELL\INSTALL* directory.

10. Open the SecureEntry workbench folder and make a copy of the **Users and groups management** object renaming it to **UCM users and groups management**. Open the new object settings notebook and change *EDYSNADM.EXE* to *UCMADM.CMD* in the path and file name entry field of the Program notebook page. Enter as parameter *EDYSNADM.EXE*

11. Make sure that the *EDYSTART.CMD* file contains the necessary Communications Manager/2 start program invocation. (usually 'start cmstart', as set up by SecureEntry/2 installation)

12. Open the UCM users and groups management object (as created above) and define in the UCM Database a userid as the administrator user.

13. Shutdown, reinitialize the workstation and sign-on with the new userid just defined.

14. You can begin now defining into the UCM Database the desired groups, group-security profiles, and resources that you wish to add to your branches. Once you have made these definitions, you will be able to implement your branch data distribution plans (Planning the distribution) in order to distribute this data to the branches.

15. Finally you can read about the utilities that UCM under OS/2 environment offers (Additional UCM Software).

Server workstations (Domain controllers)

In order to install and configure server workstations you must follow the steps explained below:

1. Customize Communications Manager/2.
2. Install SecureEntry/2 with UCM feature (without RACF, so that RACF emulator is used instead). Note that no machine reboot is required at this stage yet.
3. Copy the translation files EDYA2E.DAT and EDYE2A.DAT located in the *\SGMSHELL\INSTALL\UCMOS2* path to the *\SGMSHELL\INSTALL* directory.

4. Make sure that the *EDYSTART.CMD* file contains the necessary Communications Manager/2 start program invocation (usually 'start cmstart', as set up by SecureEntry/2 installation).
5. Shutdown, reinitialize the workstation and proceed to sign-on with a defined UCM userid.

Client workstations in the branches

To install the client workstations, you only have to install SecureEntry/2 with UCM and RACF emulator features. Remember, in addition, to specify that this is a client workstation when prompted by the installation process. Once installation has been finished you must shutdown, reinitialize the workstation and sign-on with a defined UCM userid.

Creating the UCM Database

You will find the utility *CREAUCM.CMD* located in the SecureEntry path, *INSTALL\UCMOS2* directory. You can use it as a sample to create the UCM Database under an OS/2 environment.

Use the following syntax to run this utility:

```
CREAUCM [DB_NAME] | [?]

Where

     DB_NAME: UCM Database name.
              If not specified, 'UCM' will be used by default.

     ?      : Displays help information
```

This utility has been developed as a sample. If you want to create the UCM Database with other parameters, you must change it accordingly.

Customizing Communications Manager/2

This section consists of a step-by-step list of the definitions you have to specify to upgrade your existing CM/2 configuration for the server workstation in each branch and for the UCM workstation:

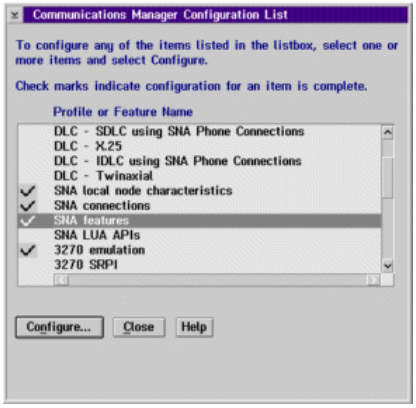
UCM workstation

Server workstations

Customizing CM/2 in the UCM workstation

This section contains the list of definitions you have to specify in the UCM workstation:

1. In the Communications Manager Configuration List panel, select **SNA features** option.

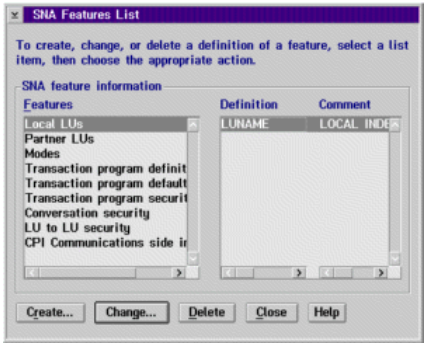


2. Create the local independent LU 6.2 that will establish the APPC connection between a server workstation and the UCM Host workstation. The LU alias must be LOCALLU.

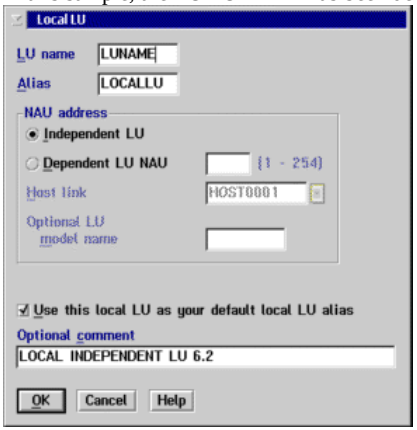
   If you already have defined one default local LU 6.2, you can go to the next step.

   If you want use a local independent LU 6.2 not configured as the default local LU, you need to use the environment variable **SGM_FORCE_LUALIAS** explained in the **SecureEntry administrator's guide**.

   This LU 6.2 is going to be referenced by the server workstations as the *partner* LU 6.2 in order to establish the UCM communication channel.



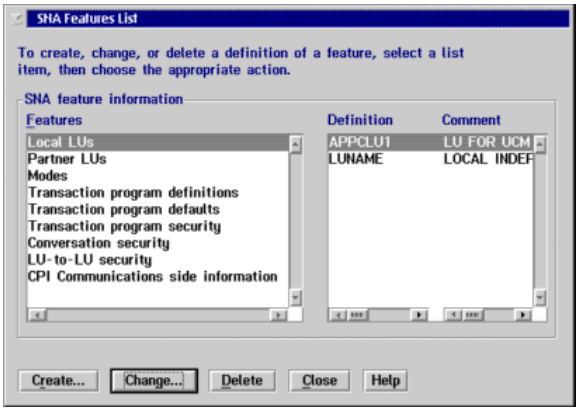
3. In this sample, the LU LUNAME has been defined as an independent LU with alias LOCALLU.



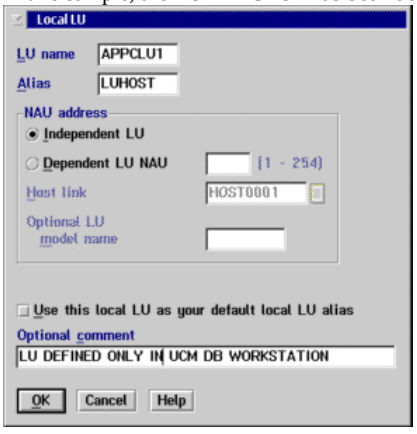
The LU name should be provided by your SNA network administrator. You can customize the LU alias following your own corporate naming rules.

In the *Additional Software* section you will find two CM/2 configuration samples in order to activate one Server workstation and the UCM workstation and to establish a communication channel between both workstations. These samples are different from the proposed configuration because these are the configuration files to establish communication between two **END nodes** without any **Network Node** presence.

4. You must define a second local independent LU 6.2, that will be used as the *partner* LU 6.2 to establish the APPC connections between the UCM workstation (when acting as a client) and itself.

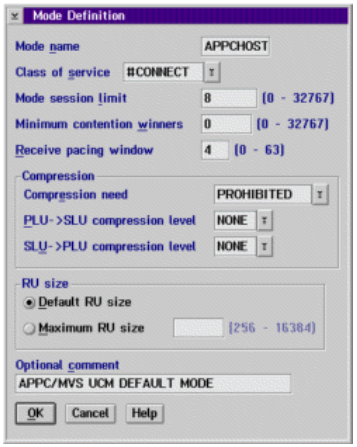


5. In this sample, the LU APPCLU1 has been defined as independent LU with alias LUHOST.

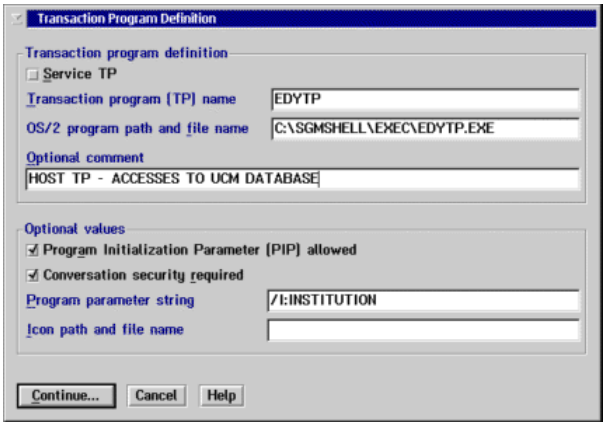


This LU name is going to be used in the CPI (Communication Side Information) definition **EDYCCHST**.

6. Add the Mode Definition with standard parameters. The only specific requirement is the Mode Name, which must be APPCHOST.

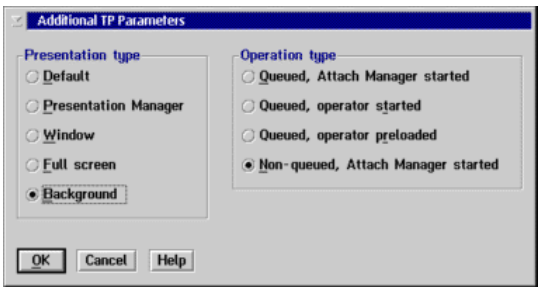


7. Define the transaction program (TP) EDYTP as explained in the following panel:



In Optional Values, select both checkboxes. As program parameters specify the Institution name for your installation (Parameter /I:INSTITUTION_NAME), that must be the same as specified during SecureEntry/2 installation, and that is stored in the file in *SENTRY.CNF* located in the *SecureEntryPath\INSTALL* directory.

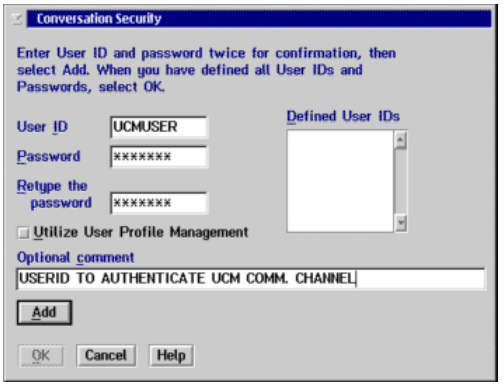
Press the **Continue** button and define in Additional TP parameters the **Presentation type** and the **Operation type** as in the following picture:



8. Define in **Conversation Security** a User ID to authenticate incoming EDYTP allocation requests.

   If you specify in the transaction program definition panel that incoming requests must supply a password and UserID, the **attach manager** will validate the supplied security before launching the EDYTP process. The attach manager will check validity of any UserID and password that arrives on incoming allocation requests. Then, the allocation request will be rejected if the UserID and password supplied does not match a valid combination in the users list.

   In the following sample the UCMUSER is defined:

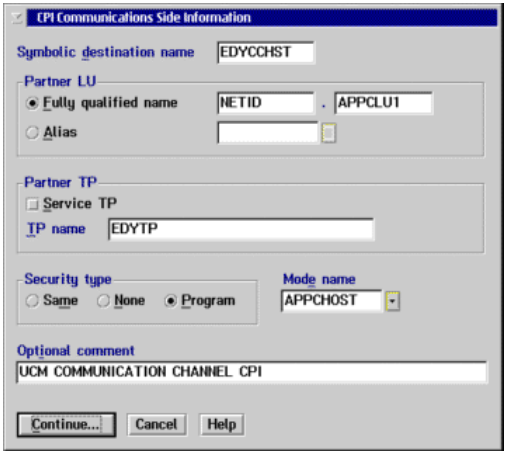


This is the UserId that you must define in the CPI (Communication Side Information) **EDYCCHST** when you select that the Security type required is *Program*.

You must remember that the UserId definition is case sensitive. If you typed an uppercase password for this UserId, you must type an uppercase password for the UserId in the CPI definition.

9. The CPI (Communications Side Information) includes the LU names and TP names that will be accessible through the CPI communication subsystem.

   You must define the EDYCCHST CPI. This CPI will be used as a communication channel between branch server workstations and the UCM Host workstation. The UCM TP name must be EDYTP and the LOGMODE used to establish the communication session must be APPCHOST. The security type must be program.



In the sample shown above this lines the TP name is EDYTP (defined in step 7), the LOG MODE used is APPCHOST (defined in step 6) and the *Partner LU 6.2* used is NETID.APPCLU1 (defined in step 5) where NETID is the network name.

In Security Type you must select the *Program* option. This causes that the session establishment is to be verified with a predefined UserId/password. Once you have selected Security type *Program* you must press the **Continue** button and then define the UserId and password.

You must specify the UserId to be used to authenticate the incoming allocation requests before launching TP EDYTP. Remember that both UserId and password must be typed as they were typed in step 8 (uppercase or lowercase), because the fields are case-sensitive.

Follows the sample dialog for user UCMUSER:

In *SecureEntryPath\INSTALL\UCMOS2* directory, you will find a Communication Manager/2 configuration file (response file) of the UCM workstation. You can activate this sample by using the instructions explained in Additional UCM Software section.

Customizing CM/2 in the Server workstations

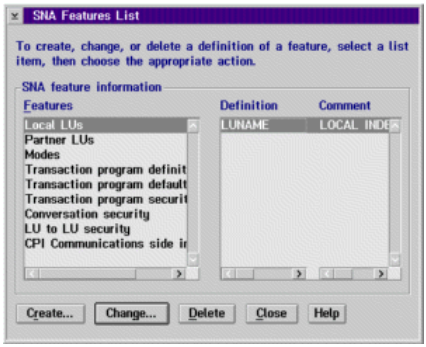
This section consists of a step-by-step list of the required CM/2 definitions for the UCM/2 branch server workstations:

1. In the Communications Manager Configuration List panel, select **SNA features** option.

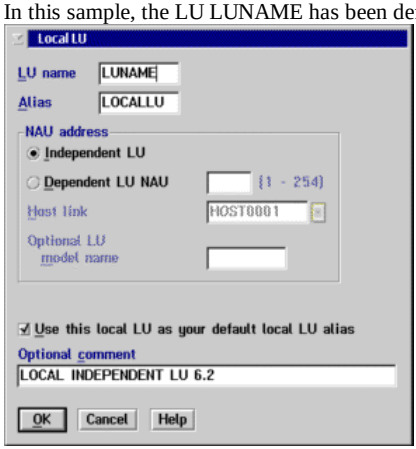
2. Create the local independent LU 6.2 that will establish the APPC connections between a server workstation and the UCM Host workstation. The LU alias must be LOCALLU.

   If you already have defined one default local LU 6.2, you can go to next step.

If you want to use a local independent LU 6.2 not configured as the default local LU, you must use the environment variable **SGM_FORCE_LUALIAS**, as explained in the **SecureEntry administrator's guide**.



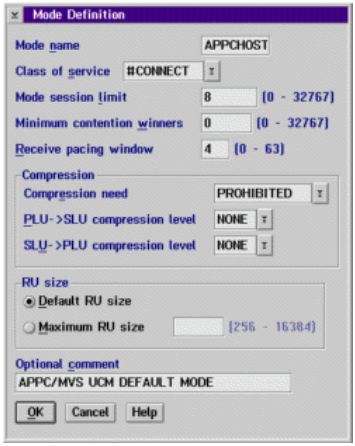
3. In this sample, the LU LUNAME has been defined as independent LU with alias LOCALLU.



The LU name should be provided by your SNA network administrator. You can customize the LU alias following your own corporate naming rules.

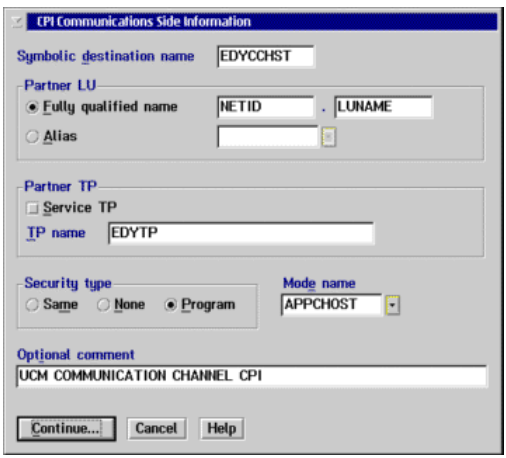
In the 'Additional Software' section you will find two CM/2 configuration samples to activate one Server workstation and the UCM workstation and to establish a communication channel between both workstations. These samples are different from the proposed configuration because these are configuration files to establish communication between two **END nodes**, without any **Network Node** presence.

4. Add the Mode Definition with standard parameters. The only specific requirement is the Mode Name, which must be APPCHOST.



5. The CPI (Communications Side Information) includes the LU names and TP names that will be accessible through the CPI communication subsystem.

   You must define the EDYCCHST CPI. This CPI will be used as the communication channel between the branch server workstations and the UCM Host workstation. The UCM TP name must be EDYTP and the LOGMODE used to establish the communication session must be APPCHOST. The security type must be program.



   In the sample shown above these lines the TP name is EDYTP, the LOG MODE used is APPCHOST (defined in step 4) and the *Partner LU 6.2* used is NETID.LUNAME (defined as default local independent LU 6.2 in the UCM workstation), where NETID is the network name.

   In Security Type you must select the *Program* option. This causes the session establishment to be verified with a predefined UserId/password. Once you have selected Security type *Program* you must press the **Continue** button and then define the UserId and password.

You must specify the same UserId as defined at step 8 of UCM workstation configuration. This user will be used to authenticate the inbound transaction before launching TP EDYTP in the UCM workstation. Remember that both UserId and password must be typed as they were in the UCM workstation because the fields are case-sensitive.

Find enclosed the sample dialog for user UCMUSER:

In *SecureEntryPath\INSTALL\UCMOS2* directory, you can find a Communications Manager/2 configuration file (response file) of the Server workstation. You can activate this sample by following the instructions explained in Additional UCM Software section.

Binding the UCM programs

SecureEntry/2 distributes the necessary software to install and customize UCM under an OS/2 environment. However, before you are able to use UCM, you must bind the UCM programs to the UCM database.

You will find the *BINDUCM.CMD* utility located in the *\SGMSHELL\INSTALL\UCMOS2* directory, which you can use in order to bind these programs.

Use the following syntax to run this utility:

```
BINDUCM [DBNAME]

Where

    DBNAME: UCM Database name under OS/2 environment. Default: 'UCM'
```

This procedure binds the UCM API. Therefore, it activates the UCM Database administration functions from the UCM Host workstation.

Remember that you will need to run this process again every time you apply service to SecureEntry/2 in this machine.

Initializing UCM Database

You will find the *UCMSETUP.CMD* utility located in the *\SGMSHELL\INSTALL\UCMOS2* directory. This utility can be used to initialize the UCM Database by filling the UCM dictionary.

Use the following command from an OS/2 window:

```
UCMSETUP [DB_NAME]
```

```
Where
      DB_NAME: UCM Database name. Default 'UCM'.
```

In addition, you will have to run the *EDYKWD.CMD* utility located in the *\SGMSHELL\INSTALL* directory, which will allow you to add the new SecureEntry/2 (not originally shipped with UCM) component names into the UCM dictionary. You can optionally defer this process (running of EDYKWD.CMD) to the moment when you set up the UCM administration function.

Once you have customized CM/2, and initialized the UCM Database, you will have to execute the **INSTSUB.EXE** program located in the *\SGMSHELL\INSTALL* directory, in order to add the supported subsystem definitions (SecureEntry/2 and UCM). Once executed successfully, you will be ready to use UCM.

Planning the distribution

To distribute branch information about resources, groups, resource-group links, and subsystems, UCM OS/2 offers the same tools as in the UCM MVS/ESA environment.

You can use the on-line distribution methods based on refresh policies, so that refresh branch data is automatically propagated to your branch servers. You will then be able to choose the refresh policy based on your corporation requirements and expectations through the UCM administrator's workstation UCM administrator's tool.

As an alternative to on-line distribution, you can use the same batch tools as available in the MVS/ESA environment. Note that in this case it will be your responsibility to distribute the branch information change packages and run the appropriate process in the branch servers.

The UCMP03 utility will build a branch information image file. With this image file you can initialize all your new branches. This method for initializing the branch data allows you to avoid communication channel information overhead whenever the branch servers connect to UCM for the first time.

The UCMP01 and UCMP02 utilities will build a file containing the UCM branch information changes in a form suitable for branch level processing.

Summarizing, you can use the following methods to distribute your branch information changes:

UCM V4RX ONLINE process

UCM V4Rx has administration tools to customize the on-line refresh function for branch information (Refresh Branch On-line). Through this tools, you can select from a list of several options when the UCM changes will be updated into the branches.

EDYSRV.EXE, a process running in LAN environment, will update this changes for every corporate branch in an automated way.

For further information about this process you can read the related chapter in the SecureEntry administrator's guide

UCM batch processes.

UCM under OS/2 environment provides with the UCMP01, UCMP02 and UCMP03 processes to gather all the branch information stored in the UCM Database. For further information of these processes you can read UCM processes (Users Centralized Management).

If you use this programs you must design two distribution plans to bring the output of the programs to the branches and update them:

PLAN1

Run the UCMP03 process to obtain all the branch information. This information will be saved in the **FILH2LAN.BIN** file, which you will be able to use to initialize a new branch. Distribute this file to the new branches and run the utility EDYUCDIS.EXE in the server workstations to activate this definitions.

PLAN2

Run UCMP01 or UCMP02 processes to gather all delta changes from the UCM Database. This information will be saved in the **DELTAH2L.BIN** file, which you will be able to use to apply the changes into your branches. Distribute this file to your existing branches and run the utility EDYUCDIS.EXE in the server workstations to activate the changes.

Note, that the users information is distributed at logon time through the UCM communications channel.

Additional UCM Software

This chapter describes the additional Software that UCM/2 installs as samples:

UCM batch procedures

ASCII/EBCDIC translation tables

Communications Manager/2 configuration sample

UCM batch procedures

With UCM under OS/2, you have available a set of batch procedures that you may need to customize and install UCM.

The following utilities are samples that you can use to develop your own procedures:

BATCHUCF.CMD

This utility concatenates the execution of the *EDYUCSRT.EXE* program (customer file preprocessor) with the *UCMPCUSF* program (customer file processor).

BATCHLOG.CMD

This is a sample of *EDYEXLOG.EXE* program that extracts data from the UCM log table.

You can find this utilities in the *\SGMSHELL\API\UCMOS2* directory.

ASCII/EBCDIC translation tables

UCM/2 provides you with the necessary passthrough ASCII/EBCDIC translation tables *EDYA2E.DAT* and *EDYE2A.DAT* that you need to install UCM/2.

These tables are located in the *SecureEntryPath\INSTALL\UCMOS2* directory. Please copy them into *SecureEntryPath\INSTALL* directory in all your **Server workstations** and also in the **UCM workstation**.

Communications Manager/2 configuration sample

In the *SecureEntryPath\INSTALL\UCMOS2* directory you will find a sample of a **Communications Manager/2** configuration for one UCM workstation and a Server workstation in a LAN Token Ring environment through adapter 0.

This configuration files were designed to work without a SNA network administrator. With this configuration files you will be able to establish a direct communication channel between two workstations (server workstation and UCM workstation).

UCMHSTWS.RSP

This is the response file to make the configuration of a **UCM workstation**. You must change the following key:

PARM_STRING

You will find this key in the TP EDYTP definition section. This is the key that EDYTP will use to descramble the user data at connection time. This is the institution name and must be the same as the corporation name defined during SecureEntry/2 installation time. time

UCMCLTWS.RSP

This is the response file to make the configuration of a **Server workstation**. You must change the following keys according to your own installation values:

DESTINATION_ADDRESS

You can find this key in **LOGICAL_LINK** section, where the link **HOSTLINK** is defined to link Server workstation with the UCM workstation.

This key must have the Token Ring node address of the UCM workstation.

From *SecureEntryPath\INSTALL\UCMOS2*, execute Communication Manager/2 setup utility as shown below in order to build both node configurations:

```
CmSetup   /r UcmHstWs.Rsp   -> In the UCM workstation


CmSetup   /r UcmCltWs.Rsp   -> In the Server workstation
```

Note that this response files have been created through **Access Feature Version 5.10**.

User Centralized Management (UCM) processes

The UCM feature enables you to create and maintain a corporate repository on the host, where consolidated corporate-scale information is stored. Information generated by the system administrator's maintenance activities as well as information corresponding to changes produced at the branches is used to modify the corporate repository. You can also use your organization existing databases to feed this repository. To do so, you must generate a customer file. The format for this customer file is described in detail in Customer file format.

To achieve the goal of security administration and user management at corporate level, UCM implements a series of processes: on-line and batch.

On-line processes

Batch processes

On-line processes

On-line processes are implemented to provide for the following situations that may involve interaction with the corporate repository. For example: some information may need to be updated or deleted, or some information may need to be retrieved and distributed.

When the system administrator carries out maintenance activities, changes are made to the corporate repository. They are directly performed from the system administrator workstation by making use of the distributed relational database architecture (DRDA).

When a user logs on to a workstation, an authentication transaction is launched and managed by the UCM Software.

When the central security subsystem repository used to authenticate users is RACF (or a similar product), the authentication transaction can be routed either through a **SNA Service TP**, or through **UCM** EDYURACF **TP**.

However, when you use UCM as your central security subsystem repository for the SecureEntry/2 users, **UCM** EDYTP **TP** is always used to manage the authentication transactions.

Both UCM *transaction programs* use a secure method to communicate users data through the UCM communications channel. When data arrives to the UCM programs, a data scrambling algorithm is used to decrypt the data.

When a user logs on to a workstation, the UCM processes search for outstanding updates of the user information on the corporate repository. If any update is found, the user data is sent to the branch to update the user profiles before the user logon is completed. This is an automatic process that helps to dynamically define or update users in the branches.

When the system administrator carries out maintenance activities for groups and resources definitions, changes are made to the corporate repository. These changes can be downloaded to the LAN branches in on-line mode. The system administrator can activate the Refresh branch on-line procedure as explained in the SecureEntry administrator's guide. This procedure downloads groups and resources change definitions from the UCM database to the LAN branches.

The system administrator can query the update levels of the branches through the Branch query application. Using this tool, the administrator will find the branches with update problems, and therefore will be able to act in order to correct the error.

The system administrator can activate the UCM Logging facility through the UCM administrator tools. This facility logs the add, update and delete operations executed against the SENT (SecureEntry) subsystem. This LOG can be managed by the EDYEXLOG batch process.

When the Refresh Branch On-line fails because of a wrong definition, you can run the EDYRVUCM recovery tool to correct it. This tool will help you to correct the error in the change tables of the UCM database. For further information about this process you can read the chapter *UCM Recovery tool* in this same manual.

UCM transaction programs

UCM provides with two different transaction programs for on-line processing of branch requests:

Transaction program EDYTP

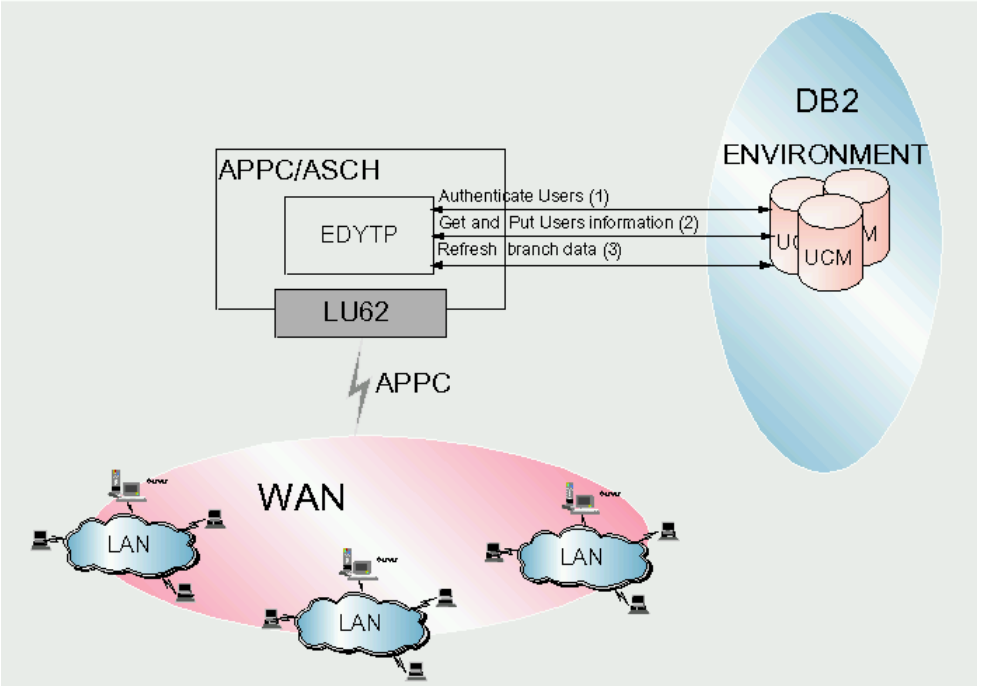
Transaction program EDYURACF

Transaction program EDYTP

MVS/ESA	OS/2	UNIX
X	X	

EDYTP is the process that processes on-line requests coming from corporate workstations at users connection time. This program gets and puts users information from/to UCM Database as needed. In addition, EDYTP is able to manage branch information, distributing UCM branch definitions to all corporate branches.

Find below a picture showing the basic UCM transactions that EDYTP can manage:



The UCM *transaction program* EDYTP has the following features:

1. Users authentication through RACF emulator

   In previous versions, the usage of the UCM program was always associated with the use of RACF authentication. Now you can customize UCM in order to manage SecureEntry/2 users authentication at logon time. If you use the RACF emulator feature, you won't have to define all your SecureEntry/2 users into the RACF repository.

   The RACF emulator has the following characteristics:

   - Users passwords are kept scrambled in UCM DB2 Database
   - The users data is scrambled before being sent through the UCM communications channel
   - Password length supported is between 4 and 14 alphanumeric characters
   - Automatic password expiration support after a defined period of time
   - Configurable logon retry limit before disabling the user account
   - Users profiles synchronization at logon/logoff times

   When a user logs on of a workstation, there is a Host transaction to synchronize (if needed) the user's profile in the LAN repository before user is allowed to work at the workstation.

   The TP EDYTP is the process that collects the user's profiles from the Host database and sends them (when needed) to the LAN environment through an *on-line* communications channel. When new profiles arrive into the LAN, they are activated before the user ends the logon process.If the user does

not exist in the LAN, it's definition is added. If the user already exists there, his LAN definition is updated.

When the user logs off a workstation, then if his personal profiles have changed, an *on-line* transaction is established in order to update the change at the Host environment. The EDYTP process receives this data and then updates it in the UCM Database.

This feature allows for users mobility across branches in your corporation. You do not have to replicate all your users database for each branch. In addition, the user will always have the feeling that he is working in the same physical workstation.

2. Unique logical view updates (Dynamic data refresh)

   The EDYTP process is able to automatically collect the UCM branch definition data and distribute it to all of your branches by replicating such data in the form of a 'branch image buffer'.

   The branch image buffer is composed of the following UCM information:

   Subsystem definitions

   Group definitions

   Resource definitions

   Group-resource, links

   User deletions

   Group-group, links

   Branch level

   Through *transaction program* EDYTP and by selecting the appropriate **refresh policy** (using the SecureEntry users and groups interactive management tool), you will be able to activate the on-line branch data distribution for your corporation, so that branch data replication occurs at the desired time. Refresh policies are fully detailed in the *SecureEntry/2 administrator's guide*.

   In addition, UCM offers a collection of batch processes for extracting from the UCM Database all the branch information needed to be updated in your corporate branches. This processes build up the branch image buffer as a file which can be distributed to all of your branches for off-line processing. This file is to be processed by a SecureEntry program (EDYUCDIS.EXE) in the LAN environment to update the branch definitions. This processes are: UCMP01, UCMP02 and UCMP03. **This is an alternate way for updating your branches without using the UCM on-line capabilities.**

You can modify EDYTP behavior/performance characteristics through the following load parameters:

By using EDYTP load parameters, you can disable EDYTP standard output, disable EDYTP output to the UCMLOG dataset, modify the refresh memory cache amount, and (only when using RACF emulation) optionally, tell EDYTP which is the INSTITUTION NAME for your corporation.

EDYTP accepts the following parameters:

/I:INSTITUTIONNAME

This parameter must be used only for those corporations having the RACF emulation feature enabled. With this parameter you can specify the INSTITUTION NAME, which should match the one specified within the SecureEntry installation tool in the branches. When you specify this parameter, the EDYTP program will not read the dataset member where this value has to be customized.

/O:nm

With this parameter you specify whether the EDYTP process must use the standard output SYSOUT (If you have it specified as DUMMY then the parameter will be ignored). Also you can specify whether the EDYTP process must log the UCM operations into the UCMLOG dataset. Both **n** and **m** can have **0** and **1** values.

n, indicates whether you want EDYTP to use/not use the standard output (SYSOUT).

m, indicates whether you want EDYTP to log/not log all UCM operations into the UCMLOG dataset.

With value **0** the feature will be disabled and with value **1** it will be enabled.

By default, the EDYTP process has both features disabled.

Note that under UCM OS/2 environments, this parameter is meaningless since EDYTP will write to the standard SecureEntry/2 traces.

/C:n

Where n indicates the primary cache buffer size. Accepted range: 10..100

When EDYTP works in multi-trans mode, it keeps an intermediate storage data buffer which will contain delta refresh information for the different update levels requested by the branches. This load parameter can be used to specify the number of different information levels that can be stored in this buffer. Increasing it may improve overall response time for on-line refresh transactions, but will also require more memory resources.

As a sample, UCM distributes a dataset member **UCM.Vxx.JCL(DATOS)** to run the EDYTP process. You must edit this member to set any required parameter for your installation. You will find below this lines a sample where the INSTITUTION NAME is specified, the UCMLOG record file is activated and memory cache increased.

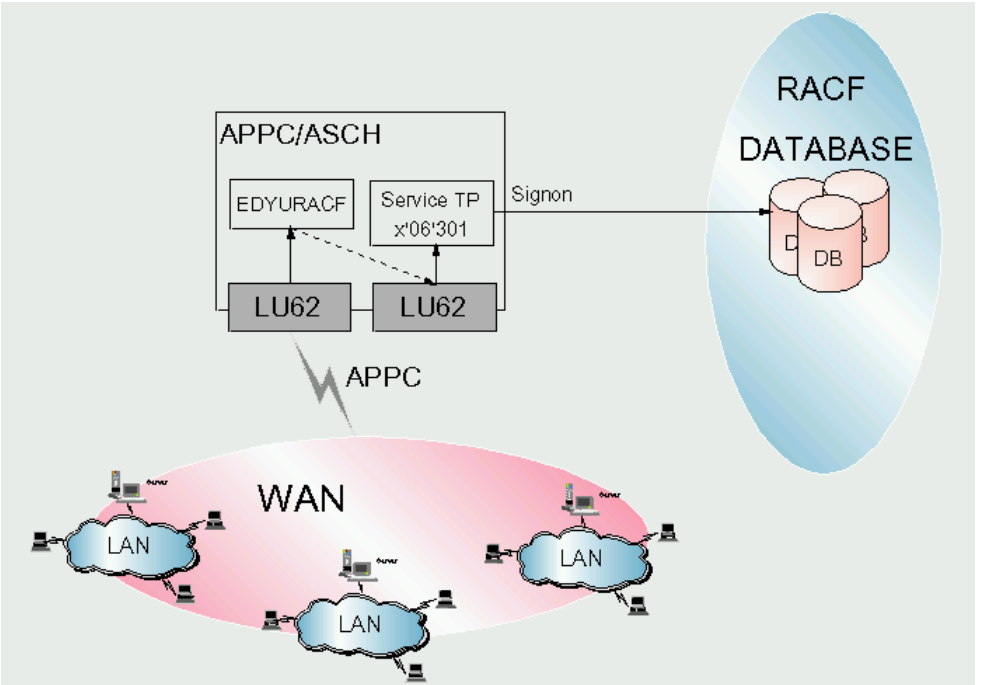
```
DSN SYSTEM(DSN)
RUN PROGRAM (EDYTP) -
    PARMS ('/I:BANCOB' '/O:01' '/C:14') -
    PLAN (PUTGETUS)
END
```

Transaction program EDYURACF

MVS/ESA	OS/2	UNIX
X		

In case you use the RACF authentication feature to validate your user's passwords, and you choose to have the communication channel information scrambled to avoid unauthorized data peeking, then you need to customize the *transaction program*EDYURACF.

EDYURACF is an APPC/MVS process that is able to receive users authentication transactions, descramble the incoming data and validate it against RACF, as shown in the following picture:



You must customize the **key** to be used by EDYURACF in order to descramble the incoming communication channel data. This **key** can be customized through a load parameter or by DD UCMINSTT:

EDYURACF load parameters

The *transaction program* EDYURACF accepts the following load parameters:

/I:decipher_key

Through this load parameter, you can inform EDYURACF which is the key to be used to descramble the incoming data. This key must be the same as used in the Server workstations to scramble the output data data (as explained below).

/O

Use this parameter to disable EDYURACF standard program output through SYSOUT (If you have SYSOUT pointing to DUMMY, this parameter does not take effect). Normally, EDYURACF output is used for problem debugging.

DD UCMINSTT

You can include in the EDYURACF JCL, the UCMINSTT DD pointing to an existing member dataset containing the descrambling key *UCMSUBSRACFKEY*. If you do not specify the **/I** load parameter, you must specify this DD. EDYURACF will search for *UCMSUBSRACFKEY = decrypt_key* in this dataset member. In case it is not found, then it will use the *INSTITUTION* key value as a default descramble key value.

Note that you have to define the *UCMSUBSRACFKEY* key in the branch server workstations in the file *EDYUCFPW.DSC* located in the *SecureEntryPath\INSTALL* directory with the same key value. If you do not

define this key, then the LAN EDYURACF component will use the *INSTITUTION NAME* as specified during SecureEntry installation to scramble the output data.

So, remember that the scramble/descramble keys have to be the same at both places: server workstations and Host environment.

Note!!

For whose customers who have already installed the beta EDYURACF *transaction program*, they can specify as scramble/descramble key the following value: **V30SCRAMBLING**. This key value forces EDYURACF to use the original (pre UCM V4x) scrambling method. Specify this value for the **UCMSUBSRACFKEY** key both in the OS/2 and MVS/ESA environments.

Please, read carefully the Installing EDYURACF section in order to obtain further information about requirements and installation steps for this transaction program.

The branch query application

MVS/ESA	OS/2	UNIX
	X	

EDYQRYBR is a program that allows you to query the levels for all branches of the corporation and store the information in one file.

This process has been designed to be used by the corporate administrator. With this process you can view all update levels of your branches. This way you can query which branches have problems and verify that all of them are at the correct level.

Note that this is an application that makes one query to the UCM central repository and extracts the actual levels for all the branches in your corporation. The level for a given branch depends on the policy refresh method that is being used within each branch server.

When one branch server updates its branch data from the UCM central repository, it does effectively download all the changes up to the actual time, and does also refresh the level indicator at the UCM repository to the previous known update level. Once this changes are applied successfully, the new level will be reported to the UCM repository at the next connection attempt. This is why the timestamps extracted from the UCM central repository for every branch will always be less than or equal to the real update level of the branches.

The syntax of this program is the following:

```
EDYQRYBR [COMMANDS:[/R:<range>] | [/T:<timestamp>]] |
         [/O:<output file>]

Where
     <range>      : n..m
     <timestamp>  : yyyy-mm-dd-hh.mm.ss
     <output file>: output of the program

Explanation

     RANGE: n..m: Range of branches to query (n <= m)
     TIMES: year-month-day-hour.minutes.seconds
            EDYQRYBR query for branches with branch level
            less or equal than specified

 If no Command is specified, EDYQRYBR will list all branches
```

```
 If option /O is not specified, EDYQRYFL.DAT will be used by default
```

This is a sample of how to query for the branches in the range of 5 to 50 and store the result information in C:\FILES\MYQUERY.DAT file:

```
Call syntax

EDYQRYBR /R:5..50  /O:C:\FILES\MYQUERY.DAT
```

UCM recovery tool (EDYRVUCM)

MVS/ESA	OS/2	UNIX
	X	

The EDYRVUCM process runs in the UCM administrator workstation. This tool can help you in correcting logical errors in the change tables of the UCM Database.

When you add an incorrect definition of one object, it remains in the change tables of the UCM Database. The refresh branch on-line process will query this tables and download the changes to the LAN branches. When the EDYSRV LAN process tries to add this definition to the LAN SERVER repository, an error is reported and the branch level is not updated to the last level of the Central Repository.

The next time that the Refresh branch on-line process is activated in the branch, the EDYSRV process will do a query for changes at the HOST with the same level and so the error will be generated again.

If the UCM administrator corrects the wrong definition, the error is purged from the UCM database, but it will remain in the change tables. This happens because the change tables store the historic operations on the UCM Database ordered by timestamp.

With EDYRVUCM process you can purge the change tables errors, but first you will have to do a process to determine which error is happening and what object is involved in it.

The mechanism to determine and correct an error is described hereafter:

1. First, you have to determine what object fails: Subsystem, Group, Resource, Links Group/Resource or Links Group/Group.

   To do this, you can peek at the EDYADMIN.LOG file. It will normally be located within the SecureEntry path, NOUSER folder of LAN branch server machines, or the UCMLOG file at the HOST environment. In the EDYADMIN.LOG file you will find all errors caused during the refresh process, and in the UCMLOG file you will only find the first error.

   When you have found the Lan Server errors you can try to correct them through the UCM administrator tool.

2. Run the EDYRVUCM process at the UCM administrator workstation with the appropriate parameters. This is, with the object found and the desired level.

   The level specifies the threshold timestamp used by the EDYRVUCM process to start the compression algorithm. You can use the min level or the max level.

   The *min* level instructs EDYRVUCM to process change tables rows for the object specified with timestamp value greater than the branch with the minor level found, that is, process all changes that are still 'alive', i.e they still have to be applied to any number of branches.

The *max* level instructs EDYRVUCM to process rows change tables for the object specified with timestamp greater than the branch with major level. That is, only changes that have never been successfully applied to any branch.

Normally the logical errors will be found with timestamp greater than the branch with major level.

The syntax of this program is the following:

```
EDYRVUCM [</O:object> & </L:level> & [/T]] | [/H]

Where

  /O[BJECT]: Is the object type to be compressed by the EDYRVUCM
             process.

  /L[EVEL] : Is the min threshold timestamp to be used by the
             EDYRVUCM process.

  /T[EST]  : Optional. Force EDYRVUCM to run in test mode.
             When you use this parameter, it is mandatory to specify
             Object and Level. With this parameter the transaction commit is
             not performed on the UCM Database.

  /H[ELP]  : Display the help screen.
```

Batch processes

UCM provides the following batch processes to enable maintenance of the corporate repository on a periodical basis.

Processes that run at the host (Host MVS or Host OS/2):

UCMPCUSF.

UCMP01.

UCMP02.

UCMP03.

EDYEXLOG.

EDYQUERY.

EDYUCSRT.

Process that runs at the LAN:

EDYUCDIS.

UCMPCUSF

MVS/ESA	OS/2	UNIX
X	X	

This process reads the customer file and the UCM repository and modifies the later with the output information. With this process you can perform the following operations on the UCM database:

ADD

The information contained in the customer file which is not present in the UCM repository is added to the database. The processing parameter for this operation is **A**.

MODIFY

The information contained in the customer file which exists in the UCM repository but has different values is updated with the values from the customer file. The processing parameter for this operation is **M**.

DELETE

The information contained in the UCM repository which is not present in the customer file is deleted from the database. The processing parameter for this operation is **D**.

FORCE

If the DELETE operation has to delete more than 10 percent of the database users, it performs a ROLLBACK instead. If you want to delete all these users, you must use the FORCE option. The processing parameter for this operation is **D F**.

The USERID column in the customer file must be sorted in ascending order. In addition, without altering this sort order, the SUBSYSTEM column must be sorted in descending order.

This is a sample of the corresponding JCL. It is distributed as data set member **UCM.Vxx.JCL(UCMPCUSF)**:

```
//STEPSORT EXEC PGM=ICEMAN
//SYSOUT   DD SYSOUT=*
//SORTIN   DD DSN=UCM.CUSTOMER.FILE,DISP=SHR
//SORTOUT  DD DSN=&&CUST,DISP=(NEW,PASS),UNIT=VIO,
//   SPACE=(CYL,(1,1)),DCB=(LRECL=1024,RECFM=FB,BLKSIZE=0)
//SORTWK01 DD UNIT=SYSDA,SPACE=(CYL,(10,10))
//SORTWK02 DD UNIT=SYSDA,SPACE=(CYL,(10,10))
//SYSIN    DD*
 SORT FIELDS=(1,10,PD,A,11,4,PD,D)
/*
//*============================================
//* (DSN COMMAND) CUSTOMER FILE
//*============================================
//DB2TSO   EXEC PGM=IKJEFT01
//STEPLIB  DD DSN=UCM.LOADLIB,DISP=SHR
//SYSUDUMP DD SYSOUT=*
//SYSPRINT DD SYSOUT=*
//SYSTSPRT DD SYSOUT=*
//UCMLOG   DD DSN=UCM.UCMLOG,DISP=SHR
//CUSTOMF  DD DSN=&&CUSTF,DISP=(OLD,DELETE)
//SYSTSIN  DD*
DSN SYSTEM(DSN)
RUN PROGRAM (UCMPCUSF) -
   PARM('A') -
   PLAN (UCMPCUSF)
END
/*
//
```

UCMP01

MVS/ESA	OS/2	UNIX
X	X	

This process gets information about UCM database (changes table) changes in:

Subsystems

Groups

Resources

Group-resource links

Users to delete

Group-group links

Level of the branches

This information is kept in a binary file that should be sent to the branches. UCMP01 gets all the information from the change tables of the UCM database and sets the transmitted flag to **Yes**. You can use this process to update the information for a branch that has been previously installed and requires the latest changes (delta information).

To run this process, you must specify the Force_Delete parameter. This parameter can take the following values:

Force_Delete=Y

The information processed and transmitted will be removed from the table of changes.

Force_Delete=N

No information will be removed.

This is a sample of the corresponding JCL. It is distributed as data set member **UCM.Vxx.JCL(UCMP01)**:

```
//DB2TSO EXEC PGM=IKJEFT01
//STEPLIB DD DSN=UCM.LOADLIB,DISP=SHR
//SYSUDUMP DD SYSOUT=*
//SYSPRINT DD SYSOUT=*
//SYSTSPRT DD SYSOUT=*
//FILH2LAN DD DSN=UCM.DELTAH2L,DISP=OLD
//SYSTSIN DD*
DSN SYSTEM(DSN)
RUN PROGRAM (UCMP01)-
   PLAN (UCMP01)-
   PARMS ('N')
   END
   /*
```

Under OS/2 environment, this process creates a file named *DELTAH2L.BIN* located in the current directory where the process is being executed.

UCMP02

MVS/ESA	OS/2	UNIX
X	X	

This process gets information about UCM database (change tables) changes in:

Subsystems

Groups

Resources

Group-resource links

Users to delete

Group-group links

Level of the branches

This information is kept in a binary file that should be sent to the branches. UCMP02 gets all the information from the tables of changes of the UCM database regardless of the transmitted flag setting and builds a file that includes all the changes. You can use this process to update the information for a branch that has been previously installed and requires all the changes (n delta information).

To run this process, no parameters are required.

This is a sample of the corresponding JCL. It is distributed as data set member **UCM.Vxx.JCL(UCMP02)**:

```
//DB2TSO EXEC PGM=IKJEFT01
//STEPLIB DD DSN=UCM.LOADLIB,DISP=SHR
//SYSUDUMP DD SYSOUT=*
//SYSPRINT DD SYSOUT=*
//SYSTSPRT DD SYSOUT=*
//FILH2LAN DD DSN=UCM.DELTAH2L,DISP=OLD
//SYSTSIN DD*
DSN SYSTEM(DSN)
RUN PROGRAM (UCMP02)-
   PLAN (UCMP02)
END
/*
```

Under OS/2 environment, this process creates a file named *DELTAH2L.BIN* located in the current directory where the process is being executed.

UCMP03

MVS/ESA	OS/2	UNIX
X	X	

This process builds a branch image getting all the needed information to initialize new branch servers. This information is kept in a binary file that should be sent to the branches.

This process gets the following information:

Subsystems

Groups

Resources

Group-resource links

Group-group links

Branch update level

This file must be processed by the *EDYUCDIS.EXE* program to initialize a new branch.

To run this process, no parameters are required.

Under OS/2 environment, this process creates a file named *FILH2LAN.BIN* located in the current directory where the program is being executed.

For those customers that install UCM under MVS/ESA, this is a sample of the corresponding JCL. It is distributed as data set member **UCM.Vxx.JCL(UCMP03)**:

```
//DB2TSO EXEC PGM=IKJEFT01
//STEPLIB DD DSN=UCM.LOADLIB,DISP=SHR
//SYSUDUMP DD SYSOUT=*
//SYSPRINT DD SYSOUT=*
//SYSTSPRT DD SYSOUT=*
//FILH2LAN DD DSN=UCM.FILH2LAN,DISP=OLD
//SYSTSIN DD*
DSN SYSTEM(DSN)
RUN PROGRAM (UCMP03)-
   PLAN (UCMP03)
END
/*
```

EDYEXLOG

MVS/ESA	OS/2	UNIX
X	X	

This process gets information from the UCM log table. This table contains information of the add, update and delete operations executed for the SENT (SecureEntry) subsystem. The operations logged in are explained in the following lines:

```
ADD     -> Add operations
UPDATE  -> Update operations
DELETE  -> Delete operations
```

This information is kept in a text file that can be processed by the customer. You can use this process to know the operations executed in the UCM database. Each line of the file describes one operation and the objects involved in.

EDYEXLOG extracts from the UCM Database the information as it is stored by DB2 in the log table. If you want to sort its output by timestamp you must use the **/S** load parameter. This parameter can take the following values:

```
/S:Y -> Output sorted by timestamp
/S:N -> Unsorted output. (default)
```

By sorting EDYEXLOG output, the amount of required memory resources will increase proportionally to the expected information volume to be processed.

EDYEXLOG reads a parameter file **DD:PARAMS**. Each register of the parameters file sets an operation to be executed by EDYEXLOG. The syntax of the **DD:PARAMS** file is explained below:

```
[EXTRACT:<operation> [OBJECT:<objecttype>]] | [DELETE] |
[PURGE] | [ROWSPROCESSED:<YES|NO>]
```

Where

- EXTRACT:<operation> [OBJECT:<object type>]
  Information to be extracted from UCM database and stored in

```
  the output file. The object to be extracted is an optional
  parameter for this line.

- DELETE
  EDYEXLOG will remove all processed rows from the UCM log table.

- PURGE
  EDYEXLOG will remove all rows below the threshold value from the UCM log
  table.

- ROWSPROCESSED
  Instructs EDYEXLOG to extract information already extracted from
  other EDYEXLOG executions. If not present, EDYEXLOG assumes NO
  by default. You can switch from NO to YES as needed.
```

When started, EDYEXLOG stores the highest timestamp of the UCM log table. This value will be used as a threshold for all following extract operations. The output from EDYEXLOG process will be stored in the file pointed by **DD:SEQLOG**.

The EDYEXLOG process stores in this file those records that match the search criteria specified in the file pointed by **DD:PARAMS** with the following record format:

Offset	Length	Description	Values
0	1	Operation type	'A'->ADD, 'U'->UPDATE, 'D'->DELETE
1	2	Object type	'U '->USER, 'G '->GROUP, 'R '->RESOURCE, 'UG'->USERGROUP, 'RU'->RESOURCEUSER, 'RG'->RESOURCEGROUP, 'GG'->GROUPGROUP
3	20	Object Id	This value depends of the operation
23	20	Object Id	This value depends of the operation
43	26	Timestamp	Operation timestamp

Note that if you are installing UCM under OS/2 environment, this program expects a parameter file named **PARAMS.OPR** to be located in the current directory where the program is being executed. Once the extract instructions have been processed, EDYEXLOG will create in the same directory an output file named **SEQLOG.OUT**.

Usage samples:

Here are two samples of the file pointed by DD:PARAMS.

We want to extract all operations from the UCM log table of the object type *User*. Then, all rows with timestamp lower than the threshold value will be removed.

```
EXTRACT:ALL OBJECT:U
PURGE
```

The same, but operation after operation:

```
EXTRACT:ADD OBJECT:U
```

```
EXTRACT:UPDATE OBJECT:U
EXTRACT:DELETE OBJECT:U
PURGE
```

We want to extract the add operations for objects *User*, *Group* and *Users/Groups links*. Also, all user add operations already processed will be extracted from UCM Database. The processed rows will be removed:

```
ROWSPROCESSED:YES
EXTRACT:ADD OBJECT:U
ROWSPROCESSED:NO
EXTRACT:ADD OBJECT:G
EXTRACT:ADD OBJECT:USERG
DELETE
```

For customers installing UCM under MVS/ESA, this is a sample of the corresponding JCL. It is distributed as data set member **UCM.Vxx.JCL(EDYEXLOG)**:

```
//DB2TSO   EXEC PGM=IKJEFT01
//STEPLIB  DD DSN=UCM.LOADLIB,DISP=SHR
//         DD DSN=DSN230.DSNLOAD,DISP=SHR
//         DD DSN=SYS1.EDC.V2R2M0.SEDCLINK,DISP=SHR
//         DD DSN=SYS1.PLI.V2R3M0.SIBMLINK,DISP=SHR
//SYSUDUMP DD SYSOUT=*
//SYSPRINT DD SYSOUT=*
//SYSTSPRT DD SYSOUT=*
//PARAMS   DD DSN=UCM.EDYEXLOG.PARAMS,DISP=OLD
//UCMLOG   DD DSN=UCM.UCMLOG,DISP=OLD
//SEQLOG   DD DSN=UCM.SEQLOG,DISP=OLD
//SYSTSIN  DD *
DSN SYSTEM(DSN)
RUN PROGRAM (EDYEXLOG) -
    PLAN (EDYEXLOG)
END
/*
```

EDYQUERY

MVS/ESA	OS/2	UNIX
X		

This process collects the value of one keyword from the UCM database and either stores it in the file pointed to by DD:UCMQUERY, or shows it on the screen.

Those corporations that install UCM under OS/2 environment will find the same feature available through EDYADMIN.CMD SecureEntry/2 administration batch utility.

When required, EDYQUERY converts the output from ASCII to EBCDIC, using a default conversion table embedded in code, or an optional user conversion table.

Syntax

To run EDYEXLOG, the object type and keyword id are specified as parameters in the form:

```
PARM('objecttype=objectid [Subsystem.]keyword /C') -
```

Where

```
objecttype: User | USERGroup | Group | GROUPGroup |
            Resource | RESOURCEGroup | RESOURCEUser
```

```
     [Subsystem.]keyword: Keyword id.
                          Can be preceded by the subsystem identifier
                          which you want to query for a keyword

     /C: Transform output from ASCII to EBCDIC.
```

example:

```
    PARM('G=SGCAS01 TREE_LOCK /C') -

    or

    PARM('U=USER0112 UCM.BRANCH') -
```

This is a sample of the corresponding JCL. It is distributed as data set member **UCM.Vxx.JCL(EDYQUERY)**:

```
//UCMQUERY JOB (0),'RUN',CLASS=A,MSGCLASS=X,MSGLEVEL=(1,1),
//         USER=UCMUSR
//*****************************************************************
//* Sample JCL to run EDYQUERY (DB2)                      UCM V4Rx *
//*                                                                *
//* If EDYA2E data set member does not exist, UCM will             *
//*  use the default translation table ASCII to EBCDIC             *
//*  coded in the program.                                         *
//*****************************************************************
//DB2TSO   EXEC PGM=IKJEFT01
//STEPLIB  DD DSN=UCM.Vxx.LOADLIB,DISP=SHR
//         DD DSN=SYS1.DSN310.SDSNLOAD,DISP=SHR
//         DD DSN=EDC220.SEDCLINK,DISP=SHR
//         DD DSN=PLI230.SIBMLINK,DISP=SHR
//SYSUDUMP DD SYSOUT=*
//SYSPRINT DD SYSOUT=*
//SYSTSPRT DD SYSOUT=*
//EDYA2E   DD DSN=UCM.Vxx.TABLES(EDYA2E),DISP=SHR
//UCMQUERY DD DSN=UCM.UCMQUERY,DISP=OLD
//UCMLOG   DD DSN=UCM.UCMLOG,DISP=OLD
//SYSTSIN  DD *
DSN SYSTEM(DSN)
RUN PROGRAM (EDYQUERY) -
    PARM('G=SGCAS01 TREE_LOCK /C') -
    PLAN (EDYQUERY)
END
/*
//
```

The dataset member **EDYA2E** is the optional conversion table (ASCII to EBCDIC). If present, it overrides the default conversion table coded in EDYQUERY module. For a description of the optional conversion table format, refer to translation table.

EDYUCDIS

MVS/ESA	OS/2	UNIX
	X	

EDYUCDIS is a program that reads the file created at the host by processes UCMP01, UCMP02, or UCMP03 and updates the server with information about:

LAN Server alias

Logon assignments

Groups (SecureEntry and LAN Server)

Security components

Branch level

...

If you run this program without parameters, you will get the on-line help. Use the following syntax to run this program:

```
path EDYUCDIS path objname level_of_messages
```

The following command is an sample of EDYUCDIS usage:

```
C:\SGMSHELL\EXEC\EDYUCDIS.EXE C:\DISTRIBU\FILEH2L.BIN 0
```

EDYUCDIS will also update the current branch level, so that if you activate the Refresh branch on-line function after executing this process, only the last changes from the UCM database will be downloaded.

EDYUCSRT

MVS/ESA	OS/2	UNIX
	X	

EDYUCSRT.EXE is an utility that can be used by customers installing UCM under OS/2 environment.

This program receives as input data a customer file created for the corporation, and creates as output a sorted customer file that can be read as input by the UCMPCUSF program.

If you run this program without parameters, a help screen will be displayed. Use the following syntax in order to run this program:

```
path EDYUCSRT /I:path\input file /O:path\output file
```

Note that if the output file is not specified, this program will create the *CUSTOMER.SRT* file in the current directory where the program is being executed. Note that this is the file name that UCMPCUSF will expect as input later on.

This is a sample to run EDYUCSRT program:

```
C:\SGMSHELL\EXEC\EDYUCSRT /I:C:\DATA\CUSTOMER.BIN
```

In this sample, EDYUCSRT receives an unsorted file named CUSTOMER.BIN and then creates a sorted output file named *CUSTOMER.SRT* that is ready to be processed, as input file, by the UCMPCUSF program.

Multi-trans transactions

If you are installing UCM under OS/2 environment, you can skip this section.

UCM allows you to configure the APPC processes in two different ways, depending on the expected workload during logon peak hours.

You can configure the UCM transaction programs in the APPC/MVS as standards TP's or as multi-trans TP's.

We recommend the use of the multi-trans configuration in those corporations having the following characteristics:

Large number of users stored in the UCM Database (>3000).

The users do logon in a small time frame interval (less than 30 minutes).

You have workload CPU problems in your installation.

When you configure an APPC process as a standard one, it is loaded for every transaction received through APPC. Once it is processed, it returns the answer to the client and is then unloaded from memory. The UCM transactions are not usually expensive transactions. Only the dynamic REFRESH branch transaction can be an expensive one. Consequently, the spent CPU time is mostly due to program memory load, unload (when the process finishes), and required system resources claiming and cleanup tasks.

If you customize the EDYTP or EDYURACF processes in multi-trans mode, the behavior will be changed as explained below:

The first transaction that arrives will load the program in memory and acquire the required resources. Once the transaction is processed and the reply sent back to the client, the process itself will not be unloaded, but will remain in memory waiting for another transaction.

The following transactions will find the process loaded in the system memory and only the normal transaction process will be executed, not requiring a new load/unload cycle.

The process will automatically be unloaded after a fixed amount of inactivity time is reached (5 minutes) without receiving further requests.

In multi-trans mode, the spent UCM CPU time by transaction is dramatically decreased during peak hours, as compared with the same process running in standard mode.

As a sample, for an installation with 3500 users the CPU time usage for the EDYTP process running in multi-trans mode has been measured to be as low as 5% of the CPU time measured with EDYTP customized in standard mode.

When the transaction program EDYTP is configured as a multi-trans process, some additional features can be activated:

Refresh memory cache to decrease DB2/MVS queries

Utilization statistics of different EDYTP transactions

You can read about EDYTP load parameters for further information.

As described for the EDYTP process, the EDYURACF TP can be customized as a multi-trans process also.

Before you customize a transaction program as a multi-trans process, we recommend that you study and measure your system workload to make sure that it is really necessary.

The configuration example that UCM provides is for EDYTP/EDYURACF transaction programs in standard mode. If you want to change this configuration to multi-trans mode, you only have to change the EDYTP profile in the APPC configuration panel.

To customize the processes as multi-trans processes, read carefully the Multi-trans customization section.

Multi-trans customization

Customizing the EDYTP and/or EDYURACF processes in multi-trans mode in APPC can be done as follows:

You must define a *RACF* user with enough privileges to execute the EDYTP/EDYURACF processes. This user will be used by the multi-trans shell. It is the generic userid of the multi-trans level. This user can be the same user as defined for the UCM channel explained in steps 8 and 9 of the Communications Manager/2 configuration section for the UCM administrator's workstation.

You have to specify that EDYTP/EDYURACF is a multi-trans process and the just defined generic userid that provides the security environment for the multi-trans shell in the *EDYTP/EDYURACF APPC profile*.

You can use the ATBSDFMU program with TPMODIFY key in order to modify the *transaction program* definition. You must modify the following values:

Change from TPSCHED_TYPE(STANDARD) to TPSCHED_TYPE(MULTI_TRANS)

Add GENERIC_ID(UCMUSER)

Optionally, if you customize the EDYTP/EDYURACF as a multi-trans process, we suggest that you use a specific execution class from APPC. The valid parameters in standard mode will still be valid in multi-trans mode. You can define a minimum and a maximum number of initiators for the EDYTP/EDYURACF, as well as the objective response time goal for the UCM transactions.

Customer file format

The file you create with the information from your existing personnel database must follow the format described in this chapter.

The customer file is a sequential database that has the following DSORG: FB,LRECL=1024,BLKSIZE=0 (sequential 1024 byte blocks under OS/2). The database records are made up of the following columns:

UserId

User name. This column is 10 characters long.

SubsystemId

Subsystem identifier. The subsystem must be previously defined in the UCM database. This column is 4 characters long. The currently available subsystems are:

SENT (SecureEntry)

UCM (User Centralized Management)

GroupId

Group to which the user belongs in the specified subsystem. The group must be previously defined in the UCM database. This column is 10 characters long.

ResourceId

Resource identifier. The resource must be previously defined in the UCM database. This column is 20 characters long.

Keyword

Every subsystem requires its own keywords. This column is 20 characters long.

ValueType

The possible types of values are:

E (EBCDIC value format)

A (ASCII value format)

B (Binary value format)

This column is 1 character long.

LengthData

It is a Short integer (2 bytes following the environment architecture OS/2 or MVS being used) that represents the value length. This column is 2 characters long.

Value

Keyword value. This column can be up to 954 characters long.

Each security subsystem supported by SecureEntry/2 and UCM need some information to be defined by each UCM user:

Subsystem required keyword information

Subsystem required keyword information

This section shows what keyword information you must specify for each of the supported subsystems (UCM and SecureEntry) and how to specify it.

UCM required keyword information

SENT required keyword information

UCM

The only required keyword for UCM is BRANCH. Its value assigns a user to a branch.

UserId	SubsystemId	GroupId	ResourceId	Keyword	ValueType	LengthData	Value
U12345678	UCM	NULL	NULL	BRANCH	E	8	filial1
U00000001	UCM	NULL	NULL	BRANCH	E	8	filial2

Each user defined in the UCM Database must have this keyword defined. You will be able to use this information later on to erase those users that do not belong to a given branch (through EDYERASE.CMD process in the LAN environment).

SecureEntry

In the SecureEntry subsystem, you can supply information for LAN Server through the keyword LAN_DATA. This keyword can have the following sub-keyword values:

Sub-keyword	**Allowed values**	**Default**
PASSWORD	A valid password	No password
PRIV_USER	GUEST, USER, and ADMIN	USER
HOME_DIR	Path name	No home dir
MAX_STORAGE	Number of KB (-1 for unlimited)	-1
DESCRIPTION	A text string up to 48 characters	Empty string
CONNECTION	0, 1	1 (allowed)
SCRIPT_PATH	Name of the script file	None
FULL_NAME	A text string up to 48 characters	Empty string
USER_EXPIRE	dd-mm-yyyy or NEVER	NEVER
HOUR_START	[0-23]	0
HOUR_END	[1-24]	24

For additional information about these LAN_DATA sub-keywords, refer to LAN Server 4.0 administration manual.

UserId	SubsystemId	GroupId	ResourceId	Keyword	ValueType	LengthData	Value
U12345678	SENT	NULL	NULL	LAN_DATA	E	hex 7EH	Sub-keywords

U12345678	SENT	Group1	NULL	NULL	NULL	NULL	NULL
U12345678	SENT	Group2	NULL	NULL	NULL	NULL	NULL

Where Sub-keywords are:

```
HOUR_START=8\0
HOUR_END=20\0
PRIV_USER=USER\0
FULL_NAME=John Smith\0
CONNECTION=0\0
USER_EXPIRE=31-12-2000\0
DESCRIPTION=Any text\0
```

The syntax for specifying sub-keywords is as follows:

```
KEYWORD=value\0
```

If you do not specify a sub-keyword, the default value is assumed.

In the customer file, NULL or \0 represent binary zeros. The LengthData field value must be the addition of the data length, the field separators (NULL value), and two null characters after the last character.

UCM problem determination guide

Accessing the UCM database may return the following messages, either by displaying them when you use the administration application at the UCM workstation or by adding entries to the UCMLOG file in MVS.

Code	Explanation	User action
UCAP 0I: Return OK.	Return code OK.	None required.
UCAP 1I: (info_string)	Informational string.	None required.
UCAP 2W: keyword (object) has an invalid size.	An update operation has been attempted, but no keywords have been specified.	Retry operation specifying one or more keywords to be modified.
UCAP 3E: Invalid ObjType: (object)	An operation with an invalid Object Type has been attempted.	Retry with a valid Object Type.
UCAP 4E: Invalid Operation: (operation)	An invalid operation has been attempted.	Retry with a valid operation.
UCAP 5W: Invalid Parameters: (parameters)	An operation with invalid parameters has been attempted.	Correct the parameters and retry.
UCAP 6E: Object (object) already exists.	An Add operation has been attempted for an object that already exists.	Correct the operation.
UCAP 7E: Object (object) does not exist.	An operation that requires an object did not found it.	Create the object or change operation.
UCAP 8E: Mandatory Keys Expected: (keys).	Missing mandatory key in the requested operation.	Correct the operation.
UCAP 9E: Keys not expected: (keys).	Unexpected key found in operation.	Correct the operation.
UCAP 10E: Key (key) does not exist.	An operation that requires a key did not found it.	Correct the operation.
UCAP 11E: Key (key) exists.	A key already existed.	Correct the operation.
UCAP 12E: Key (key) is incorrect.	A key has been found incorrect.	Correct the operation.
UCAP 13E: Not all keywords in (object)	Missing keywords in the operation.	Correct the operation.
UCAP 14I: Many rows fetched. Number rows: (rows).	Too much information has been retrieved.	Concrete the search operation.
UCAP 15I: No rows fetched in (object)	No information is available.	None required.
UCAP 16E: User delete in (subsystem) subsystem not allowed.	A deletion of a required user in a subsystem has been attempted.	Change operation.
UCAP 17E: Subsystem delete (subsystem) not allowed.	A deletion of a required subsystem has been attempted.	Change operation.
UCAP 18E: Default does	A default is not defined in the	Grant a keyword value.

not exist: (keyword).	t_keyword table for that keyword.	
UCME errno E: (error string)	An internal allocation function has failed.	Check system memory.
UCSQ sqlcode E: (sql string)	An SQL statement has failed.	Check the sqlcode in the DB/2 Problem Determination Guide.

In addition, UCM writes information in a log file, EDYUCM.LOG, which you can find on every branch server workstation. The entries in the log file can have one of the following formats:

Information message

```
12/11/96 12:04:30 (Comp) [I] Text
```

Warning or error message

```
12/11/96 12:04:30 (Comp) [[W]|[ERR]] Text
                                  (Comp) rc=+retcode
```

Error message

```
12/11/96 12:04:30 (Comp) [ERR] Text
                                  (Comp) rc=+retcode
                                  Component exited.
```

where *Comp* can be NETBIOS, EDYSRV, and EDYFREE

You can read also the following messages list:

NETBIOS messages

EDYSRV messages

EDYFREE messages

NETBIOS messages

Code	Explanation	User action
NETBIOS [I] Netbios reset. Using adapter (adapternum).		
NETBIOS [I] Netbios name (servername) added.		
NETBIOS [W] The resources allocated are S=(sessionsnumber), C=(commandsnumber), N=(namesnumber). NETBIOS rc=+56		
NETBIOS [ERR] [Listening | Receiving | Sending | Hanging]	A fatal error has caused EDYSRV to unload.	Check the Netbios return code.

NETBIOS rc=+(netbios return code) Component exited.		
NETBIOS [ERR] The resources allocated are S=(sessionsnumber), C=(commandsnumber), N= (namesnumber). NETBIOS rc=+56 Component exited.	Insufficient Netbios resources have caused EDYSRV to unload.	Check the Netbios resources.
NETBIOS [ERR] Requested Netbios resources not available. NETBIOS rc=+56 Component exited.	Insufficient Netbios resources have caused EDYSRV to unload.	Check the Netbios resources.
NETBIOS [ERR] Maximum number of applications exceeded. NETBIOS rc=+54 Component exited.	Insufficient Netbios resources have caused EDYSRV to unload.	Check the Netbios resources.
NETBIOS [ERR] The server Netbios Name (servername) is not unique. NETBIOS rc=+(netbios return code) Component exited.	A duplicate Netbios name has caused EDYSRV to unload.	Check the Netbios return code.

EDYSRV messages

Code	Explanation	User action
EDYSRV [I] Component unloaded after FREE request.		
EDYSRV [ERR] Invalid number of sessions. Component exited.	No number of sessions specified in loading parameter.	Retry with a valid number of sessions.
EDYSRV [ERR] Invalid adapter number. Component exited.	Invalid adapter number specified in loading parameter.	Retry with a valid adapter number.
EDYSRV [ERR] Invalid server name. Component exited.	Invalid server name specified in loading parameter.	Retry with a valid server name.
EDYSRV [ERR] Invalid	An invalid value has been specified as loading parameter.	Retry with a valid value.

loading parameter. Component exited.		
EDYSRV [ERR] Error loading ACSNETB.DLL. Component exited.	The Netbios run-time library has not been loaded.	Check that Netbios is correctly installed.
EDYSRV [ERR] Insufficient memory available. Component exited.	Not enough memory available.	Check system memory.
EDYSRV [ERR] The password is not valid in LS. EDYSRV rc=+1	Password not valid in Lan Server.	Check password.
EDYSRV [ERR] Password is not valid in RACF. EDYSRV rc=+1	Password not valid in RACF.	Check password.
EDYSRV [ERR] Password expired. EDYSRV rc=+2	Password is expired in LS.	Change password.
EDYSRV [ERR] Password is expired. EDYSRV rc=+2	Password is expired in RACF.	Change password.
EDYSRV [ERR] Invalid password parameter . EDYSRV rc=+3	Password invalid in LS.	Change password.
EDYSRV [ERR] EDYSRV rc=+4	Timeout in an UCM logon or logoff process.	Check communications and retry.
EDYSRV [ERR] User has not been found in LS: (userid). EDYSRV rc=+7	The specified Userid is not defined in Lan Server.	Define Userid and retry.
EDYSRV [ERR] User (userid) has not been found in RACF. EDYSRV rc=+7	The specified Userid is not defined in RACF.	Define Userid and retry.
EDYSRV [ERR] EDYSRV rc=+7	User is not defined in UCM.	Define Userid and retry.
EDYSRV [ERR] New password is incorrect in RACF. EDYSRV rc=+9	The password is not valid in RACF.	Change password and retry.
EDYSRV [ERR] The user is not allowed to logon at this time. EDYSRV rc=+10	The user has a time restriction to logon.	Wait to logon.
EDYSRV [ERR] Racf General Security Error. Ret_code: 7 EDYSRV rc=+11	A non authorized user has attempted to sign on to RACF.	Retry with an authorized user.
EDYSRV [ERR]	User account disabled in Lan Server.	Enable account.

EDYSRV rc=+12		
EDYSRV [ERR] EDYSRV rc=+13	User account expired in Lan Server.	Update account.
EDYSRV [ERR] Invalid Opercode. EDYSRV rc=+61	An invalid code operation has been specified to EDYSRV.	Correct the operation code and retry.
EDYSRV [ERR] EDYSRV rc=+61	EDYSRV internal error.	Unload EDYSRV and load again.
EDYSRV [ERR] Insufficient memory to process request. EDYSRV rc=+61	EDYSRV internal error.	Check system memory.
EDYSRV [ERR] Thread creation has failed. EDYSRV rc=+61	EDYSRV internal error.	Check system resources.
EDYSRV [ERR] LAN Server has failed. Ret_code: (RetCode). EDYSRV rc=+62	An operation to LAN Server has failed.	Check LAN Server RetCode.
EDYSRV [ERR] Racf General Security Error. Ret_code: 6 EDYSRV rc=+63	An incorrect data format has been passed to RACF.	Check data format and RACF configuration.
EDYSRV [ERR] EDYSRV rc=+64	General UCM error in logon or logoff process.	Check communications. Start SENTRY traces.
EDYSRV [ERR] Memory error in UCM Database EDYSRV rc=+65	EDYTP internal error.	Check assigned memory for EDYTP process at HOST environment. Check communications. Start SENTRY traces.
EDYSRV [ERR] EDYSRV rc=+66	General error in WSOD refresh profiles process.	Check communications. Start SENTRY traces.
EDYSRV [ERR] EDYSRV rc=+67	WSOD refresh profiles did not find profiles to refresh.	

EDYFREE messages

Code	Explanation	User action
EDYFREE [I] EDYFREE has been invoked.		

Appendix.

This section contains the explanation of UCM features that can not be placed in a separate chapter.

Translation table.

UCMLOG file. Information, execution problems and statistics

How can I improve UCM (EDYTP) performance?

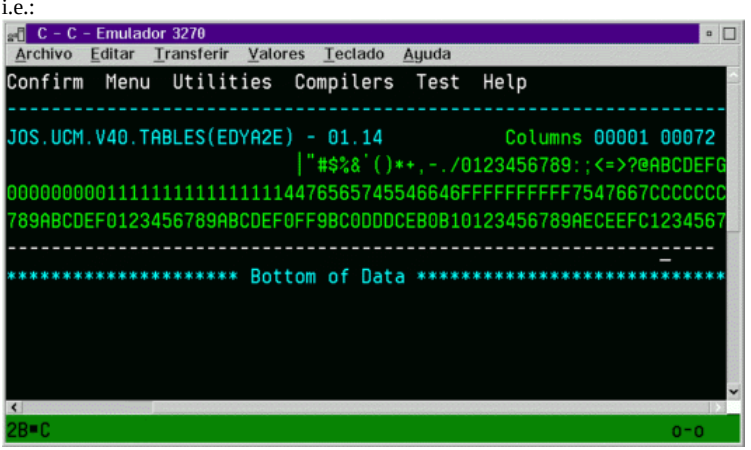
ASCII/EBCDIC translation table

As SecureEntry V3.0 does, UCM V4Rx allows for the user to code an optional translation table to override the default ASCII/EBCDII conversion made by the UCM modules.

ASCII to EBCDIC table format.

This table is a dataset member with a unique conversion record 256 bytes long. The position of every byte, 0 to 255 (0 to FF hexadecimal), represents the ASCII character. The value coded on every position represents the equivalent EBCDIC character.

i.e.:



On the distributed sample **UCM.Vxx.TABLES(EDYA2E)**, the column 65 (in a 0-255 range) has a EBCDIC character A. That means that the ASCII A (decimal 65, hexadecimal 41) will be converted to EBCDIC A (hexadecimal C1).

Comment records are allowed. They must have an * in column 0 and an * in column 255 (always on a 0-255 range).

UCM V4Rx supplies one sample dataset member **UCM.Vxx.TABLES(EDYA2E)** as a guide to users having to use their own conversion table. As distributed, EDYA2E is equal to default table embedded on UCM modules. So, if you do not need to make any changes, there is no reason to use this sample table at all.

If you are installing the UCM host function under a OS/2 environment, remember to use the supplied pass-through translation tables instead, as explained in the installation chapter.

UCMLOG file. Information, execution problems and statistics

Under MVS/ESA environment, most UCM processes will write information into a dataset pointed by **DD:UCMLOG**.

UCM processes can write three different information record types into the UCMLOG dataset:

```
- Informative
  Information about normal execution of the UCM processes,
  to inform you about the operations made against UCM Database.

- Errors
  Information about severe errors happened while the UCM processes were
  running. When this errors are related to DB2 operations, the full
  SQLCA structure is written into the UCMLOG dataset.

- Statistical
  Statistic information about processed data by the UCM programs. This
  information can be very useful for the EDYTP process in order to know
  the number of transactions executed by your branches and as a way to
  detect possible bottlenecks.
```

In multi-trans mode, each EDYTP instance awakened by APPC will keep its information in this dataset. you can differentiate an EDYTP instance from another through its EDYTP ID, which is composed by the following information:

```
ID=XXXXXXXX-JJJ-hh:mm:ss

Where

   XXXXXXXX = hexadecimal number 8 length
   JJJ      = Execution Julian day
   hh:mm:ss = Hour, minute and second when EDYTP was started.
```

Once EDYTP is unloaded from memory by APPC, it will write its statistics into the UCMLOG dataset, as shown in the following sample:

```
-EDYTP (ID=47BB0698-316-04:30:17): ----- MULTI-TRANS RESULTS -----
-EDYTP (ID=47BB0698-316-04:30:17): SIGNON TX OK              = 1234
-EDYTP (ID=47BB0698-316-04:30:17): SIGNON TX NOK             = 0
-EDYTP (ID=47BB0698-316-04:30:17): UCM LOGON  TX OK          = 622
-EDYTP (ID=47BB0698-316-04:30:17): UCM LOGON  TX NOK         = 0
-EDYTP (ID=47BB0698-316-04:30:17): UCM LOGOFF TX OK          = 13
-EDYTP (ID=47BB0698-316-04:30:17): UCM LOGOFF TX NOK         = 0
-EDYTP (ID=47BB0698-316-04:30:17): GET BRANCHID TX OK        = 4
-EDYTP (ID=47BB0698-316-04:30:17): GET BRANCHID TX NOK       = 0
-EDYTP (ID=47BB0698-316-04:30:17): REFRESH TX OK             = 259
-EDYTP (ID=47BB0698-316-04:30:17): REFRESH TX NOK            = 0
-EDYTP (ID=47BB0698-316-04:30:17): REFRESH AVERAGE (SECONDS) = 0.023166
-EDYTP (ID=47BB0698-316-04:30:17): REFRESH GETS DB2 DATA     = 4
-EDYTP (ID=47BB0698-316-04:30:17): PRIMARY REFRESH CACHE
-EDYTP (ID=47BB0698-316-04:30:17): LEVEL <1999-11-10-22.07.13.253638>. BRANCHES WITH
THIS LEVEL <233>. BUF. SIZE <2158>
```

```
-EDYTP (ID=47BB0698-316-04:30:17): LEVEL <1999-11-09-22.08.46.201035>. BRANCHES WITH
THIS LEVEL <12>. BUF. SIZE <4254>
-EDYTP (ID=47BB0698-316-04:30:17): LEVEL <1999-04-12-12.45.17.137138>. BRANCHES WITH
THIS LEVEL <10>. BUF. SIZE <50102>
-EDYTP (ID=47BB0698-316-04:30:17): LEVEL <1999-11-08-22.07.35.579767>. BRANCHES WITH
THIS LEVEL <1>. BUF. SIZE <5564>
-EDYTP (ID=47BB0698-316-04:30:17): SECONDARY REFRESH CACHE
-EDYTP (ID=47BB0698-316-04:30:17): TOTAL TX PROCESSED = 2128
-EDYTP (ID=47BB0698-316-04:30:17): FINISHING NORMALLY

 Where

 - For each transaction type, EDYTP shows those that have finished
   successfully and those that have finished with errors.

 - For refresh transactions, EDYTP shows how many times it
   failed obtaining data from its memory cache (thus causing DB2 accesses),
   and how many different buffers were stored. For each stored buffer,
   it shows how many branches did request for this buffer.

 - Finally, EDYTP shows the total number of transactions processed
   while it was running.
```

Under OS/2 environment, the UCM processes keep their problem determination information in the regular SecureEntry/2 traces. Since the multi-trans feature is only available for MVS/ESA environments, no statistical information will be available.

How can I improve UCM performance?

If you have installed UCM under MVS/ESA and you wish to improve the UCM transactions performance, you must read carefully this section:

Activate multi-trans feature for EDYTP process.

If you are using the on-line refresh feature, you could customize it at 'FIXED DAY TIME'. This policy distributes better the resources that your branches require at the Host. Try to choose a time with low workload at the host site.

If you are using on-line refresh feature, you can increase the minutes interval to be used to calculate the init hour of refresh branch, thus lowering the chances that all your branches attempt to refresh at the same time.

You can modify the number of primary elements kept in memory cache for EDYTP. This memory is used to keep branch data to be sent to branches, avoiding expensive DB2 accesses.

If you have not yet installed UCM, think about the possibility of using the RACF emulator. Through this feature you will decrease the number of transactions done against APPC during users connection time.

If you are using the SYSPRINT output feature you can override it through an EDYTP load parameter. This output normally can be used for debug tasks.

If you are using the UCMLOG feature, you can override it, as explained above, through an EDYTP load parameter. Only severe errors and statistics (if you are using multi-trans mode) will be kept in UCMLOG dataset.

If you are using the RACF emulator feature you can use an EDYTP load parameter in order to inform it about your INSTITUTION NAME.

If you have installed EDYURACF TP you can improve its response time by:

Activating the multi-trans feature for EDYURACF TP.

If you are using the SYSPRINT output feature, you can override it through an EDYURACF load parameter. This output is normally used only for debug tasks.

Use a load parameter to specify the data scrambling key to be used.

If you keep on having performance problems, check your installation to see if you can:

Increase APPC/MVS execution priority,

Increase the number of initiators that can be used by the UCM transaction programs,

Decrease the response time goal for UCM TP's, or

Contact with your VTAM/APPC administrators.

WHAT'S NEW !!

This chapter is intended as an 'easy guide' for upgraders. It lists in chronological, descending order, the links to the appropriate documentation points where new function has been added to the product. Note that for a full reference of fixes and additions, we encourage you to look into the README file supplied within the first installation diskette image of SecureEntry/2, since the ones shown below are only what we consider main improvements/changes.

New *transaction program* EDYURACF to authenticate users against RACF.

Now UCM can be installed under OS/2 environment.

New Multi-trans mode to improve throughput for UCM on-line transactions.

New RACF emulator feature.

New EDYRVUCM utility to correct and compress the change tables of the UCM Database.

New EDYQRYBR utility to query all the update levels of the corporation branches.

New chapter about Migrating from earlier UCM versions.

New UCM logging facility explained in EDYEXLOG and EDYQUERY processes.

Optional user ASCII to EBCDIC translation table for EDYQUERY.

New method to update groups and resources change definitions explained in On-line processes