Customization Guide

CL/SUPERSESSION®
CL/GATEWAY®

Version 147

G251238100

September 1996

Candle Corporation
2425 Olympic Boulevard
Santa Monica, California 90404

Contents

List of Illustrations

List of Tables

Preface

About this document

This manual provides instructions and explanations for customizing CL/SUPERSESSION®, CL/GATEWAY® for MVS, and CL/GATEWAY for IMS for your network, system, and users. Before using this manual, be sure you have completed all steps in the following documents:

- *Program Directory*
- *Basic Configuration Guide*

Customer Support

Introduction

Candle provides electronic support and telephone support to assist you when you have questions about Candle products. Customer support is available 24 hours a day, seven days a week.

Electronic support

Candle Electronic Customer Support® (CECS℠) enables you to search for existing questions, answers, problems, and fixes; review preventive service planning (PSP) information; and open incidents for Candle products. CECS is available through the Advantis™ network and by direct PC dial-up. For registration information, call your nearest Candle Support Services office.

Telephone support

If you have an urgent problem or need to talk to a Candle Support Services representative, contact the Support Services office nearest you.

Office	Telephone	Fax
North America		
Santa Monica	(800) 328-1811 (310) 829-5844	(310) 582-4204
Europe		
Antwerp	(32) (3) 272-3606	(32) (3) 272-3607
Breda	(31) (76) 520.19.09	(31) (76) 520.19.19
Duesseldorf	(49) (21) 193-6920	(49) (21) 193-69220
Manchester	(44) 161 499 3503	(44) 161 437 5225
Munich	(49) 89 54 5540	(49) 89 54 5541-19
Paris	(33) (1) 5361 6000	(33) (1) 5361 0515
Sollentuna	(46) 8 623 1235	(46) 8 623 1855
Asia Pacific		
Hong Kong	(852) 2528 6289	(852) 2865 0770
Kuala Lumpur	(603) 230 9930	(603) 230 9932
Singapore	(65) 220 50 92	(65) 226 35 79
Sydney	(61) 2 9954 1500	(61) 2 9954 1818
Tokyo	(81) 3 5562-6991	(81) 3 5562-6995

International customers

When your local support office is unavailable, you may contact Candle's North America support center. If USADirect® service is available in your country, use the 800 telephone number. If USADirect service is not available, ask your international operator for assistance in calling Candle's local (310) number.

Incident information

A Candle support representative may request the following information when you call to report a problem:

- your Candle personal ID (PID) number
- the release level of the Candle product
- the release level of IBM® or other vendor software
- identifying information and dates of recently applied maintenance to the Candle product
- a detailed description of the problem (including the error message) and what led up to the failure
- a description of any unusual events that occurred before the problem

Incident documentation

You may be asked to send incident documentation to Candle Support Services. On the outside of the package, please write the incident number given to you by the Candle Support Services representative. Send your documentation addressed as follows:

Candle Support Team
Candle Support Center, *incident number*
2425 Olympic Boulevard
Santa Monica, California 90404

Documentation Conventions

Introduction

Candle documentation adheres to accepted typographical conventions for command syntax. Conventions specific to Candle documentation are discussed in the following sections.

Panels and figures

The panels and figures in this document are representations. Actual product panels may differ.

Revision bars

Revision bars (|) may appear in the left margin to identify new or updated material.

Variables and literals

In examples of command syntax, uppercase letters are actual values (literals) that the user should type; lowercase letters are used for variables that represent data supplied by the user. Default values are underscored.

```
LOGON APPLID(cccccccc)
```

In the above example, you type **LOGON APPLID** followed by an application identifier (represented by *cccccccc*) within parentheses. The application identifier can have at most eight characters.

Note: In ordinary text, variable names appear in italics.

Symbols

The following symbols may appear in command syntax.

Symbol	Usage
|	The 'or' symbol is used to denote a choice. Either the argument on the left or the argument on the right may be used. Example: `YES | NO` In this example, YES or NO may be specified.
[ ]	Denotes optional arguments. Those arguments not enclosed in square brackets are required. Example: `APPLDEST DEST [ALTDEST]` In this example, DEST is a required argument and ALTDEST is optional.

{ }	Some documents use braces to denote required arguments, or to group arguments for clarity. Example:
	
	`COMPARE {workload} -`
	`        REPORT={SUMMARY | HISTOGRAM}`
	
	The *workload* variable is required. The REPORT keyword must be specified with a value of SUMMARY or HISTOGRAM.

_	Default values are underscored. Example:
	
	`COPY infile outfile -`
	`    [COMPRESS={YES | NO}]`
	
	In this example, the COMPRESS keyword is optional. If specified, the only valid values are YES or NO. If omitted, the default is YES.

ƀ	The symbol ƀ indicates a blank space, when needed for clarity.

Documentation Set

Introduction

Candle provides a complete set of documentation for CL/SUPERSESSION and CL/GATEWAY. Each manual in this documentation set contains a specific type of information to help you use the product.

Candle welcomes your comments and suggestions for changes or additions to the documentation set. A user comment form, located at the back of each manual, provides simple instructions for communicating with Candle's Technical Documentation department.

Product documentation

The documentation listed in the following table is available for CL/SUPERSESSION and CL/GATEWAY. To order additional product manuals, contact your Candle Support Services representative.

Table 1 (Page 1 of 2). CL/SUPERSESSION and CL/GATEWAY Documentation

Document Number	Document Name	Description
LS60-3779	Version 147 Release Guide	Contains new information for this release.
LS99-3783	Program Directory	Provides installation instructions and details all other installation considerations.
LS55-3785	Basic Configuration Guide	Provides basic instructions for customizing CL/SUPERSESSION and CL/GATEWAY to the specific needs of your network, system, and users.
LS51-3781	Customization Guide	Provides instructions and explanations for customizing CL/SUPERSESSION and CL/GATEWAY to the needs of your network, system, and users.
LS54-3786	User's Guide	Contains brief instructions on how to operate CL/SUPERSESSION and CL/GATEWAY.
LS99-3789	Operator's Guide	Describes the CT/Engine operator facility and commands used by CT/Engine, CL/SUPERSESSION, and CL/GATEWAY.
LS99-4225	Introduction to SSPL Dialogs	Introduces users to the Structured Session Procedure Language (SSPL); shows how to customize and use some simple dialogs written in SSPL.
LS99-3821	SSPL Programming Guide	Explains how to create your own dialogs with SSPL, using a sample application that creates and manages a table.
LS53-3787	Dialog Language Reference Manual	Contains comprehensive descriptions of all features of the SSPL dialog language.

Table 1 (Page 2 of 2). CL/SUPERSESSION and CL/GATEWAY Documentation

Document Number	Document Name	Description
LS57-3780	Problem Determination Guide	Contains instructions and documentation recommendations for locating and solving problems in CL products.
LS52-3788	Messages Manual	Lists and explains all CT/Engine, CL/SUPERSESSION, and CL/GATEWAY messages and suggests appropriate user actions.
LVM99-4103	Quick Reference Card	Pocket-sized document that contains step-by-step instructions for using CL/SUPERSESSION and CL/GATEWAY.
LS59-3801	Master Index	Contains a master index for all CL/SUPERSESSION and CL/GATEWAY manuals that contain indexes.

Chapter 1. Options for Setting Up Your Network

This chapter explains the options for setting up your network with CL/SUPERSESSION and CL/GATEWAY. You will learn how to

- specify entry point dialogs
- connect users to a network entry point
- create a VTAM® resource definition for CL/GATEWAY and CL/SUPERSESSION
- APF-authorize the VTAM path
- identify the new names of the product libraries and find ACB names in those libraries

Network Entry Points

A *network entry point* establishes an interface to a CT/Engine™ entry point dialog for physical terminal users. You can use an entry point dialog to make decisions about the user who is logging on. You might decide, for example, that one user should get multisession access, another should get single-session access, and another should not be allowed any access at all.

When a user logs onto an application identified by a HOSTGATE command, the dialog identified in the command receives control. (Refer to "Logon Dialog Sequence" on page 235 for an overview of the dialog sequence.) More than one network entry point can be defined in a single CT/Engine execution environment (that is, MVS address space). However, each must reference a unique VTAM APPL (known as *entry point*, *entry point application*, or *entry point configuration*).

You specify the CL/GATEWAY entry point dialog in the HOSTGATE command in *&rhilev*.RLSCMDS (KLGCHGGW). By default, the CL/GATEWAY entry point dialog is KLGATEWY.

How Terminals Connect to CL/SUPERSESSION or CL/GATEWAY

You can use four methods for connecting users to a network entry point:

- the VTAM interpret table
- the VTAM LOGAPPL
- the CT/Engine FORWARD command in conjunction with a VTAM LOGAPPL
- SIMLOGON from CLSDST PASS applications

The method you choose depends on how you want users to access the system and what you want to present to them.

VTAM LOGAPPL

You can specify the VTAM LOGAPPL parameter to connect a terminal automatically with a network entry point. If you use the VTAM LOGAPPL parameter with CL/SUPERSESSION or CL/GATEWAY to connect your terminals, VTAM automatically logs the terminal onto CL/SUPERSESSION or CL/GATEWAY.

Under this method, a user who logs off or disconnects a session with CL/SUPERSESSION or CL/GATEWAY can get back to VTAM and select another entry point or application.

CT/Engine FORWARD Command with the VTAM LOGAPPL

The FORWARD command, used together with a VTAM LOGAPPL, lets you connect a user directly to a network entry point and lock the user into that entry point. You can specify a temporary entry point as the target of the LOGAPPL, and the FORWARD command routes the logon to the permanent network entry point.

For example, to forward users to the KLGICFG1 entry point, follow these steps:

1. During installation you copied *&thilev*.TLSSAMP(KLSVTLST) into SYS1.VTAMLST(*newname*). Member *newname* contains the APPL statements that define the applications to VTAM. This member is considered the *major node*.

   Add the following line to SYS1.VTAMLST(*newname*):

   ```
   FORWARD APPL ACBNAME=KLVF1,AUTH=(ACQ,NVPACE,PASS)
   ```

   FORWARD is the APPL name, and KLVF1 is the ACB name for a dummy APPL from which the FORWARD command routes a user to the selected entry point.

2. Add this command to the end of *&rhilev*.RLSCMDS(KLSSTART):

   ```
   FORWARD KLVF1 applname
   ```

   where *applname* is the application netname of the selected entry point.

   Note: If used, FORWARD should always be the last command coded in KLSSTART to ensure that all commands are executed.

3. Use the VTAM LOGAPPL to connect terminals to FORWARD.

4. To activate the FORWARD APPL:

   a. Shut down the CT/Engine address space.
   b. Inactivate and then reactivate the *newname* major node.
   c. Restart the CT/Engine address space.

Alternatively, you can issue a temporary FORWARD command without recycling the CT/Engine address space:

1. Add the FORWARD APPL to a new major node with a unique name.
2. Activate the new major node.
3. Issue the FORWARD command from the CT/Engine operator facility.

   Note: If you use the temporary method, the FORWARD command remains in effect only until the next time CT/Engine is shut down.

Because users are forwarded immediately from the LOGAPPL entry point, they cannot return to VTAM unless an operator deactivates the LOGAPPL.

You can avoid using the VTAM LOGAPPL by issuing the FORWARD command with the LOGAPPL operand, but Candle does not recommend doing so. The FORWARD LOGAPPL generates a queued logon, which conflicts with the APPLDEF SIMLOGON queued logon (see “SIMLOGON from CLSDST PASS Applications”). Without the LOGAPPL parameter, the FORWARD command is compatible with the APPLDEF SIMLOGON.

For detailed information on the FORWARD command, see the *Operator's Guide*.

SIMLOGON from CLSDST PASS Applications

When a user connects to a CLSDST PASS application, the session between the user's terminal and CL/GATEWAY terminates. When the session with the CLSDST PASS application ends, the user's physical terminal either returns to VTAM control, or is passed back to the gateway or to a LOGAPPL. If the gateway requires user ID and password validation, the user must reenter this information to get back to the Main Menu after logging off an application.

With the SIMLOGON parameter of the APPLDEF command, you can automatically reestablish a session with CL/GATEWAY after termination of a session with a CLSDST PASS application. In addition, if the gateway requires user ID and password validation, you can use operands of the SIMLOGON parameter to pass this information to CL/GATEWAY when the session is reestablished. From the user's perspective, control returns directly to the Main Menu when the application terminates.

Warning:

You cannot use both the APPLDEF SIMLOGON and the FORWARD LOGAPPL for the same application. The FORWARD LOGAPPL generates a queued logon, which conflicts with the SIMLOGON queued logon. Without the LOGAPPL parameter, the FORWARD command is compatible with the APPLDEF SIMLOGON.

VTAM Resource Definitions for CL/GATEWAY and CL/SUPERSESSION

CL/GATEWAY and CL/SUPERSESSION require the definition of a VTAM application major node (VBUILD) and a series of logical units (APPLs) to support the following:

- **CT/Engine Operator Facility**

  The CT/Engine operator facility supports monitoring and control functions for authorized personnel. Full CT/Engine operator capabilities are available from an MVS console. Optionally, you can define an APPL to provide CT/Engine operator functions from any terminal. In the distributed SYS1.VTAMLST, in *&thilev*.TLSSAMP(KLSVTLST), the default ACB name for the CT/Engine operator facility is *&lsvt0*, and the network name is VTPOPER.

  You can also define an APPL to activate the VTAM Program Operator (VPO) and allow CT/Engine operators to issue VTAM commands. In SYS1.VTAMLST(*newname*), the default ACB name for the VPO is *&lsvt4*, which is defined in *&rhilev*.RLSPARM(KLVINVPO).

- **Entry Points**

  A CL/GATEWAY entry point command, found in *&rhilev*.RLSCMDS(KLGCHGGW), establishes an application access path into the network. Each entry point requires a unique APPL and can support many users.

  Network entry points (also known as configurations) provided for CL/GATEWAY include KLGICFG1, KLGICFG2, and KLGICFG3 (ACB names *&lsvt1*,*&lsvt2*, *&lsvt3* respectively).

  Note: Be sure to set a high EAS value for the entry point applications; for example, if you expect to have 2,500 users on KLGICFG1, set EAS=2500 in the *&lsvt1* APPL in SYS1.VTAMLST(*newname*).

- **CUA™ Operator**

  The CUA operator provides an interface for operator commands and functions through pull-down menus and pop-up windows. In the distributed SYS1.VTAMLST, its default ACB name is *&lsvt5*, and its network name is CUAOPER.

- **IMS/DC Virtual MTOs (CL/GATEWAY for IMS only)**

  IMS/DC MTO virtual sessions implement CL/GATEWAY for IMS session services. Each IMS/DC MTO virtual terminal requires one APPL. In SYS1.VTAMLST, the ACB names for the virtual MTOs are *&lsmto1* through *&lsmto4*.

- **Virtual Terminals**

  Virtual terminal pools consist of one or more APPL-defined logical units used by CL/GATEWAY and CL/SUPERSESSION to establish virtual sessions with VTAM application programs. Each virtual terminal requires one APPL. In SYS1.VTAMLST(*newname*), the ACB names and network names for the virtual terminals are *&lsvt00* through *&lsvt70*.

Specifying an Entry Point for LU1 Devices

The entry points provided with CL/GATEWAY and CL/SUPERSESSION do not work with LU1 devices. LU1 devices need their own entry points, which cannot be multisession and cannot use table displays. Table services can be used. The LU1 cannot use PSM. Furthermore, their configuration cannot specify KLSLMOD as the logmode panel.

If any of your users have LU1 terminals, follow this procedure to create a network entry point for them.

1. Copy *&thilev*.TLSSAMP(KLGLU1SO) and *&thilev*.TLSSAMP(KLGLU1PW), shown in Figure 1 and Figure 2 on page 25, into *&rhilev*.RLSPNLS and modify them for your environment.

```
)OPTION LEVEL(1)
)COPY KLSATTR1

)DECLARE
n        scope(local)             * N(o) password change
q        scope(local)             * Question mark
ovtype   scope(local)             * Variable ovtype contains carriage
msgarea  scope(local)             * Message area
appl     scope(share)             * Desired application
viguser  scope(session)           * Userid
vigpswd  scope(session)           * Password

)PROLOGUE

/*********************************************************************
* Variable ovtype contains carriage return characters,               *
* followed by characters to overtype the password,                   *
* followed by a line feed.                                           *
*********************************************************************/
set ovtype '\0D##########\0D**********\0D%%%%%%%%%%\15'
set n 'N'                         /* N(o) password change          */
set q '?'                         /* Question mark                 */
set appl ''                       /* Null out the application      */

)BODY
Enter userid:                       _viguser =
Enter current password:             %vigpswd   =&ovtype
Enter desired application:          _appl    =
Do you want to change your password&q_n=

&msgarea

)EPILOGUE

if &syskey = ''
   set syskey 'ENTER'

if !&vsplang
   set vsplang '1'

/*********************************************************************
* Select a dialog to change password on request.                     *
*********************************************************************/
if &n ne 'N'                          /* If requesting to change    */
   dialog 'KLGLU1PW'                   /* password - goto new        */
                                      /* password panel.            */
```

Figure 1 (Part 1 of 2). Sample Logon Panel for LU1 Devices

```
if !&appl do                           /* If no desired appl,        */
   set msgarea 'Enter Desired Application'
   set viguser ''                      /* let the user start over    */
   set vigpswd ''                      /* again.  Enter userid/pwsd  */
   reshow                              /* Reshow this panel          */
end

/*********************************************************************
*  Set resflag to 1 will bypass userid/password resolution in GNTRY   *
*********************************************************************/
set resflag 11                         /* Set resflag to 1            */
/*********************************************************************
*  Strip leading blanks, fold to UC, blank OK as end delimiter.       *
*********************************************************************/

if '&vigpswd' do                       /* If password specified,     */
   set vigpswd fold(LJUST('&vigpswd' 8)) /* Uppercase and            */
   set i (INDEX('&vigpswd',' '))       /* remove leading blanks      */
   if &i >= 0
      set vigpswd (SUBSTR('&vigpswd',0,&i))
   set vigpswd '&encdec('&vigpswd')'   /* Encrypt it                 */
end

/*********************************************************************
*  Strip leading blanks, fold to UC, blank OK as end delimiter.       *
*********************************************************************/

if '&viguser' do
   set viguser fold(LJUST('&viguser' 8))
   set i (INDEX('&viguser',' '))
   if &i >= 0
      set viguser (SUBSTR('&viguser',0,&i))
end

/*********************************************************************
*  Strip leading blanks, fold to UC, blank OK as end delimiter.       *
*********************************************************************/

if (set rc (VALIDATE('&viguser' '&encdec('&vigpswd')')) eq 4 or
   (VALIDATE('&viguser' '&encdec('&vigpswd')'))) eq 8 do
   set vigrnpsw ''
   select gntry
end
else if &rc = 0 do
   set vigrnpsw ''
   select gntry
end
else do
   set msgarea '&vtpdfval'
   set vigpswd ''
   set viguser ''
   reshow
end

)TERM
set msgarea  ''
```

Figure 1 (Part 2 of 2). Sample Logon Panel for LU1 Devices

```
)OPTION LEVEL(1)
)COPY KLSATTR1
)DECLARE
i        scope(local)              * Index placeholder
ovtype   scope(local)              * Overtype characters
vigpswd1 scope(local)              * New password
vigpswd2 scope(local)              * Verify new password
vignpswd scope(share)              * New password
msgarea  scope(local)              * Message area
vsplang  scope(share)              * NLS language, default to 1
syskey   scope(share)              * SYSKEY

)PROLOGUE

/*********************************************************************
* Variable ovtype contains carriage return characters,               *
* followed by characters to overtype the password,                   *
* followed by a line feed.                                           *
*********************************************************************/
set ovtype '\0D#########\0D*********\0D%%%%%%%%%\15'

)BODY
Enter new password:                 %vigpswd1  =&ovtype
Verify new password:                %vigpswd2  =&ovtype

&msgarea

)EPILOGUE

if &syskey = ''
   set syskey 'ENTER'

if !&vsplang
   set vsplang '1'
```

Figure 2 (Part 1 of 2). Sample New Password Entry for LU1 Users

```
/*******************************************************************
* Process the ENTER key.                                          *
*******************************************************************/
if &syskey = 'ENTER' do               /* Enter key pressed         */
   set vigpswd1 fold(LJUST('&vigpswd1' 8)) /* strip leading blanks   */
   set i (INDEX('&vigpswd1',' '))
   if &i >= 0
      set vigpswd1 (SUBSTR('&vigpswd1',0,&i))
   set vigpswd2 fold(LJUST('&vigpswd2',' '))
   set i (INDEX('&vigpswd2',' '))
   if &i >= 0
      set vigpswd2 (SUBSTR('&vigpswd2',0,&i))
   if ('&vigpswd1' eq '') or ('&vigpswd2' eq '') do
      set vigpswd1 ''
      set vigpswd2 ''
      set msgarea 'New password must be entered'
      reshow
   end
   if '&vigpswd1' ne '&vigpswd2' do
      set vigpswd1 ''
      set vigpswd2 ''
      set msgarea 'New password must be verified'
      reshow
   end
   set vignpswd '&encdec('&vigpswd1')'
   if '&vignpswd' do
      set vigpswd1 ''
      set vigpswd2 ''
      set msgarea 'Password changed'
      return
   end
end

)TERM
set msgarea  ''
```

Figure 2 (Part 2 of 2). Sample New Password Entry for LU1 Users

2. If KLGATEWY is not in *&rhilev*.RLSPNLS, copy *&thilev*.TLSPNLS(KLGATEWY) into *&rhilev*.RLSPNLS. Add the following code to the KLGATEWY dialog, right after the **LOGON_USER:** label:

```
If &appl     /* Is this an LU1 device? */
  pass &appl (&syslmode '&viguser '&encdec('&vigpswd')'')
```

3. Create a new gateway configuration member in *&rhilev*.RLSPARM. Copy *&rhilev*.RLSPARM(KLGICFG1) into the new configuration member, and globally change KLGLGON to KLGLU1SO.

4. In *&rhilev*.RLSCMDS(KLGCHGGW) add a HOSTGATE command to define a new CL/GATEWAY entry point. Associate this HOSTGATE command with the new gateway configuration member and with the broadcast group of the LU1 users.

   For detailed information on the HOSTGATE command, see the *Operator's Guide*.

VTAM Authorized Path

CT/Engine, CL/SUPERSESSION, and CL/GATEWAY support the VTAM authorized path when they run APF-authorized. The resulting performance improvements are passed on to all users and may be particularly valuable for virtual sessions.

In TLVLOAD concatenation, all libraries must be APF-authorized (that is, identified in the IEAAPFxx member of SYS1.PARMLIB). The libraries as distributed contain the proper linkage editor SETCODE values.

The CT/Engine initialization stream must specify APF=Y. A default initialization stream for CT/Engine is provided in *&rhilev*.RLSPARM(KLSSYSIN). When STEPLIB concatenation is APF-authorized, KLSSYSIN defaults to APF=Y. When the VTAM authorized path is used, you may include the SRBEXIT=YES operand in any APPL statement that defines the CT/Engine operator, entry point dialogs, or virtual terminals.

Note: You may specify SRBEXIT=YES only if CT/Engine is APF-authorized.

Product Libraries

In Version 147 of CL/SUPERSESSION and CL/GATEWAY, dataset names and member names were changed. Table 2 lists the old and new names of five of the product's primary libraries, along with the common name of each library and a brief description of each.

Three of the previous libraries (VTPCLIB, VTPILIB, and VTPPLIB) have been separated into a CL/SUPERSESSION library and a CT/Engine library. Each CL/SUPERSESSION library name begins with TLS, and each CT/Engine library name begins with TLV.

Note: Each library name will have a high-level qualifier that is determined at installation time.

Table 2 (Page 1 of 2). CL/SUPERSESSION Libraries

Old Name	New Name	Common Name	Description
user.VTPCLIB	*&rhilev*.RLSCMDS	Command Library	Command lists that control initialization and operation
VTPCLIB	TLVCMDS	TLSCMDS	
user.VTPILIB	*&rhilev*.RLSPARM	Initialization Library	Initialization and configuration parameters
VTPILIB	TLVPARM	TLSPARM	
user.VTPPLIB	*&rhilev*.RLSPNLS	Panel Library	Dialogs that control the user interface panels (or *panels*)
VTPPLIB	TLVPNENU	TLSPNLS	
LSAMPLE	TLSSAMP	Sample Library	Sample dialogs, utilities, and exits

Table 2 (Page 2 of 2). CL/SUPERSESSION Libraries			
Old Name	**New Name**	**Common Name**	**Description**
VTPSLIB	TLVSPENU	Sample Library	Sample dialogs, utilities, and exits

&thilev.TLSSAMP(KLSTBLS) contains a cross-reference list of the old and new names of all data elements that have changed.

Where to Find the ACB Names in the Product Libraries

To help you customize without causing ACB names to conflict, Table 3 shows the locations of ACB name definitions provided with CL/GATEWAY and CL/SUPERSESSION.

Table 3. ACB Names Provided with CL/GATEWAY and CL/SUPERSESSION			
ACB Name	**Library**	**Member**	**Application**
&lsvt0	*&rhilev*.RLSCMDS	KLSSTART	CT/Engine operator facility
&lsvt1 *&lsvt2* *&lsvt3*	*&rhilev*.RLSCMDS	KLGCHGGW	CL/GATEWAY entry points, defined in HOSTGATE commands
&lsvt5	*&rhilev*.RLSCMDS	KLSSTART	CUA Operator
&lsvt4	*&rhilev*.RLSPARM	KLVINVPO	VTAM Program Operator (VPO)
&lsmto1 through *&lsmto4*	*&rhilev*.RLSCMDS	KLICIMTO	CL/GATEWAY for IMS virtual MTOs, defined in VSM commands
&lsvt00 through *&lsvt70*	*&rhilev*.RLSCMDS	KLS$VSMS	Virtual terminals

CL/SUPERSESSION Started Task JCL and Runtime Datasets

The started task JCL and runtime datasets for CL/SUPERSESSION follow. Use the started task JCL to start the CL/SUPERSESSION address space.

```
//*********************************************************************
//*
//*  PROCEDURE:  KLS - EXECUTE CL/SUPERSESSION
//*
//*  FUNCTION:
//*
//*     THIS SAMPLE PROCEDURE MAY BE USED TO START CL/SUPERSESSION.
//*     THE USER MAY TAILOR THIS JCL TO SATISFY INSTALLATION
//*     DATASET NAMING AND SYSOUT HANDLING REQUIREMENTS.
//*     CL/SUPERSESSION CAN BE RUN AS A BATCH JOB OR AS A STARTED TASK.
//*
//*  NOTES:
//*
//*     THE STEPLIB DD STATEMENT IS REQUIRED ONLY IF THE CL/ENGINE
//*     PRODUCT DOES NOT RESIDE IN A LINKLIST LIBRARY.
//*
//*     THE TLVLOAD DD STATEMENT MUST REFER TO THE EXECUTION LIBRARY.
//*
//*     IF AN EXTERNAL SECURITY SYSTEM WILL BE USED TO PERFORM
//*     NETWORK ACCESS VALIDATION, THE EXECUTION LIBRARY MUST BE
//*     APF AUTHORIZED.
//*
//*     IT IS RECOMMENDED THAT THE EXECUTION LIBRARY IS
//*     APF AUTHORIZED.
//*
//*     &THILEV
//*       IS THE SMP TARGET HILEVEL QUALIFIER
//*     &RHILEV
//*       IS THE RUN-TIME NON-VSAM HILEVEL QUALIFIER
//*
//*********************************************************************
//KLS      PROC PFX='&THILEV',
//             RPFX='&RHILEV',
//             MEMORY=0M,         CL/SUPERSESSION REGION ALLOCATION
//             SYSIN=KLSSYSIN,    INIT LIBRARY STARTUP MEMBER
//             SOUT=A             LOG AND DEBUGGING OUTPUT CLASS
//*
//IEFPROC  EXEC PGM=KLV,REGION=&MEMORY.TIME=1440
//STEPLIB  DD   DISP=SHR,DSN=&PFX..TLVLOAD
//TLVLOAD  DD   DISP=SHR,DSN=&RPFX..RLSLOAD
//         DD   DISP=SHR,DSN=&PFX..TLSLOAD
//         DD   DISP=SHR,DSN=&PFX..TLVLOAD
//TLVCMDS  DD   DISP=SHR,DSN=&RPFX..RLSCMDS
//         DD   DISP=SHR,DSN=&PFX..TLSCMDS
//         DD   DISP=SHR,DSN=&PFX..TLVCMDS
//TLVPARM  DD   DISP=SHR,DSN=&RPFX..RLSPARM
//         DD   DISP=SHR,DSN=&PFX..TLSPARM
//         DD   DISP=SHR,DSN=&PFX..TLVPARM
//TLVPNLS  DD   DISP=SHR,DSN=&RPFX..RLSPNLS
//         DD   DISP=SHR,DSN=&PFX..TLSPNLS
//         DD   DISP=SHR,DSN=&PFX..TLVPNENU
//TLVH0ENU DD   DISP=SHR,DSN=&PFX..TLVHPENU
//TLVLOG   DD   SYSOUT=&SOUT
//TLVSNAP  DD   SYSOUT=&SOUT
//TLVSYSIN DD   DISP=SHR,DSN=&RPFX..RLSPARM(&SYSIN),FREE=CLOSE
//ABNLIGNR DD   DUMMY            * TO TURN ABENDAID OFF *
```

Figure 3. CL/SUPERSESSION Version 147 Started Task JCL

The following DD statements are required.

STEPLIB The CL/SUPERSESSION and CT/Engine load module library may be specified via STEPLIB or may optionally be placed in the MVS system LINKLIST. If it is placed in the LINKLIST, the STEPLIB is not necessary.

APF authorization is required if any of the following is true:

- The ACF/VTAM™ authorized path facility is used.
- NAF records are written to SMF.

- An external security system is used to authorize users.
- MVS, VTAM, or IMS commands are issued from the CT/Engine operator facility.

TLVLOAD

As with STEPLIB, the TLVLOAD DD statement also refers to the CL/SUPERSESSION and CT/Engine load library. This library contains all CL/SUPERSESSION load modules, plus any installation written exits.

Note: The TLVLOAD dataset accessed through the TLVLOAD DD statement requires the same APF authorization as the STEPLIB load library.

See “CL/ENGINE Product Libraries Reference” on page 245 for the syntax to be used when editing this library.

TLVCMDS

The TLVCMDS DD statement identifies the command list library. Command lists are processed automatically during CT/Engine initialization and can be invoked from an MVS console or CT/Engine operator terminal.

See “CL/ENGINE Product Libraries Reference” on page 245 for the syntax to be used when editing this library.

TLVPARM

The TLVPARM DD statement identifies the initialization library that contains various members that configure particular product features and functions, such as operator capabilities and VIEWLOG.

See “CL/ENGINE Product Libraries Reference” on page 245 for the syntax to be used when editing this library.

TLVPNLS

The TLVPNLS DD statement identifies the panel library that defines all panels for CL/SUPERSESSION and CT/Engine users.

See the *Dialog Language Reference Manual* for information about coding SSPL dialogs.

TLVH0ENU

The TLVH0ENU DD statement identifies the help library.

TLVLOG

The TLVLOG DD statement defines the datsaset collecting CL/SUPERSESSION log output. DCB attributes are set internally. The blocksize defaults to 6144 but may be overridden by a startup parameter.

This dataset provides a printed record of all CL/SUPERSESSION activity, including optionally invoked debugging information.

TLVSNAP

The TLVSNAP DD statement defines a sequential datsaset to collect a CL/SUPERSESSION formatted SNAP in the event of CL/SUPERSESSION abnormal termination.

TLVSNAP DCB attributes are specified within CL/SUPERSESSION to conform to MVS SNAP macro requirements.

	Note: Do not code DCB attributes on the TLVSNAP DD statement.
TLVSYSIN	This dataset is read by CL/SUPERSESSION at system initialization. It specifies the initialization library member (KLSSYSIN) that contains system startup parameters. Recommended initialization parameters are distributed in RLSPARM. See "CL/ENGINE Product Libraries Reference" on page 245 for the syntax to be used when editing this library.
ABNLIGNR	If ABEND-AID™ is installed, this DD statement disables it so that usable CL problem documentation is produced for Candle Support Services.

The PROC parameters are as follows:

PFX	The high-level prefix of CL/SUPERSESSION target dataset names. In this example, the high-level prefix is &THILEV.
RPFX	The high-level prefix of CL/SUPERSESSION runtime dataset names. In this example, the high-level prefix is &RHILEV.
SYSIN	The member name within RLSPARM that contains CT/Engine initialization parameters. In this example, the name is KLSSYSIN.
MEMORY	The CT/Engine address space region requirements. In this example, the region requirements are 0M.
SOUT	The SYSOUT class name. In this example, the name is A.

Chapter 2. Virtual Terminals

A physical terminal is usually allowed only one session at a time, so CT/Engine creates *virtual terminals*. Virtual terminals simulate physical terminals for sessions between users and applications, but each virtual terminal can support many sessions on behalf of many users, and each user can have sessions on more than one virtual terminal.

When a user selects a SINGLE or MULTI application, CT/Engine allocates a virtual terminal, which logs onto the application for the user. This results in a *virtual session*.

VTAM APPL statements define virtual terminal logical units to create virtual sessions. From the user's perspective, a virtual session between a terminal and an application is the same as a standard terminal-to-application session. However, a virtual session actually consists of two distinct physical sessions:

- one session between the user and CT/Engine
- one session between CT/Engine and the destination application

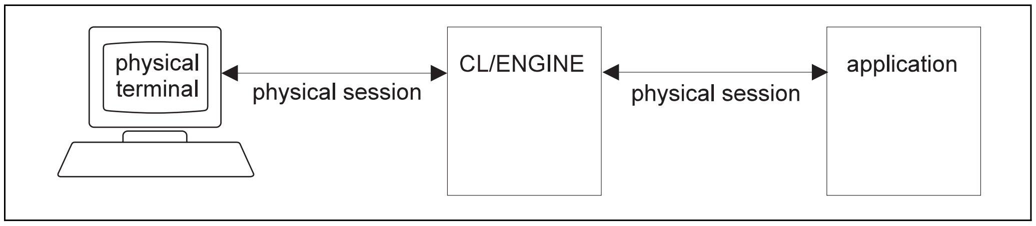

Figure 4. CL/SUPERSESSION Virtual Session

During installation you copied *&thilev*.TLSSAMP(KLSVTLST) into SYS1.VTAMLST(*newname*). Member *newname* contains the APPL statements that define the applications to VTAM. This member is considered the *major node*, and the logical units defined by APPL statements are the *minor nodes*.

Each virtual terminal requires one APPL. For complete information on the APPL statement, refer to the IBM publication *VTAM Installation and Resource Definition*. Some of the APPL statements included in the *&thilev*.TLSSAMP major node member KLSVTLST look like those shown in Figure 5 on page 34.

KLSVTLST contains the STS1.VTAMLST APPL definitions that are required to implement the products distributed on the tape.

If you change the APPL names or ACB names, be sure to change the appropriate commands in *&rhilev*.RLSCMDS. The KLVINVPO member in *&rhilev*.RLSPARM is the only member affected by KLSVTLST.

EAS=1 is recommended for all virtual terminals not supporting parallel sessions. These virtual terminals will not have enough active sessions to warrant the half- or full-page of (E)CSA used by VTAM as a session hash-table when EAS is specified as or defaults to greater than 44.

Copy KLSVTLST into SYS1.VTAMLST and activate it before attempting to run the sample configurations.

```
        VBUILD TYPE=APPL
******************************************************************
*     CL/ENGINE OPERATOR FACILITY. REFER TO MEMBER KLSSTART IN
*     DDNAME TLVCMDS FOR THE COMMANDS THAT ACTIVATE THE
*     OPERATOR ACBS.
******************************************************************
&lsvt0  APPL  AUTH=(ACQ,PASS,NVPACE),PARSESS=YES
******************************************************************
*     CL/ENGINE CUA OPERATOR FACILITY. REFER TO MEMBER KLSSTART IN
*     DDNAME TLVCMDS FOR THE COMMANDS THAT ACTIVATE THE
*     OPERATOR ACBS.
******************************************************************
&lsvt1  APPL  AUTH=(ACQ,PASS,NVPACE),PARSESS=YES
******************************************************************
*     CL/ENGINE VTAM PROGRAM OPERATOR FACILITY.
*     REFER TO MEMBER KLVINVPO IN DDNAME TLVPARM.
******************************************************************
&lsvt2  APPL  AUTH=(ACQ,PASS,SPO)
******************************************************************
*     SAMPLE SUPERSESSION/GATEWAY APPLICATIONS.  REFER TO
*     KLGCHGGW IN DDNAME TLVCMDS FOR THE HOSTGATE COMMANDS.
******************************************************************
&lsvt1  APPL  AUTH=(ACQ,PASS,NVPACE),PARSESS=YES
&lsvt2  APPL  AUTH=(ACQ,PASS,NVPACE),PARSESS=YES
&lsvt3  APPL  AUTH=(ACQ,PASS,NVPACE),PARSESS=YES
******************************************************************
*      VIRTUAL TERMINAL DEFINITIONS REQUIRED FOR IMS.
*      REFER TO MEMBER KLICIMTO IN DDNAME TLVCMDS FOR THE
*      VSM COMMANDS.
*
*      &lsmto1 THRU 4 ARE THE VIRTUAL mto LU'S.
******************************************************************
&lsmto1 APPL  EAS=1,AUTH=(ACQ,NVPACE)
&lsmto2 APPL  EAS=1,AUTH=(ACQ,NVPACE)
&lsmto3 APPL  EAS=1,AUTH=(ACQ,NVPACE)
&lsmto4 APPL  EAS=1,AUTH=(ACQ,NVPACE)
******************************************************************
*     VIRTUAL TERMINAL DEFINITIONS FOR POOL VIRTPARS.
*     SEE MEMBER KLS$VSMS IN DDNAME TLVCMDS FOR THE
*     VSM COMMANDS.
******************************************************************
&lsvt00 APPL AUTH=(ACQ,NVPACE),PARSESS=YES
******************************************************************
*     VIRTUAL TERMINAL DEFINITIONS FOR POOLS VIRTDED AND IINPOOL.
*     SEE MEMBER KLS$VSMS IN DDNAME TLVCMDS FOR THE
*     VSM COMMANDS.
******************************************************************
```

Figure 5 (Part 1 of 3). APPL Statements in KLSVTLST

```
&lsvt01 APPL AUTH=(ACQ,NVPACE),PARSESS=YES
&lsvt02 APPL AUTH=(ACQ,NVPACE),PARSESS=YES
&Lsvt03 APPL AUTH=(ACQ,NVPACE),PARSESS=YES
&lsvt04 APPL AUTH=(ACQ,NVPACE),PARSESS=YES
&lsvt05 APPL AUTH=(ACQ,NVPACE),PARSESS=YES
&lsvt06 APPL AUTH=(ACQ,NVPACE),PARSESS=YES
&lsvt07 APPL AUTH=(ACQ,NVPACE),PARSESS=YES
&lsvt08 APPL AUTH=(ACQ,NVPACE),PARSESS=YES
&lsvt09 APPL AUTH=(ACQ,NVPACE),PARSESS=YES
&lsvt10 APPL AUTH=(ACQ,NVPACE),PARSESS=YES
*******************************************************************
*     VIRTUAL TERMINAL DEFINITIONS FOR POOLS
*       VIRTPASS  11 THROUGH 70
*       VIRT3270  11 THROUGH 70
*       MODEL2    11 THROUGH 70
*       MODEL3    11 THROUGH 70
*       MODEL4    11 THROUGH 70
*       MODEL5    11 THROUGH 70
*       MODEL9    11 THROUGH 70
*       HCFPOOL   11 THROUGH 70
*       TSOESA    11 THROUGH 20
*       TSOPOOL   11 THROUGH 20
*     SEE MEMBER KLS$VSMS IN DDNAME TLVCMDS FOR THE
*     VSM COMMANDS.
*******************************************************************
&lsvt11 APPL AUTH=(ACQ,NVPACE),EAS=1
&lsvt12 APPL AUTH=(ACQ,NVPACE),EAS=1
&lsvt13 APPL AUTH=(ACQ,NVPACE),EAS=1
&lsvt14 APPL AUTH=(ACQ,NVPACE),EAS=1
&lsvt15 APPL AUTH=(ACQ,NVPACE),EAS=1
&lsvt16 APPL AUTH=(ACQ,NVPACE),EAS=1
&lsvt17 APPL AUTH=(ACQ,NVPACE),EAS=1
&lsvt18 APPL AUTH=(ACQ,NVPACE),EAS=1
&lsvt19 APPL AUTH=(ACQ,NVPACE),EAS=1
&lsvt20 APPL AUTH=(ACQ,NVPACE),EAS=1
&lsvt21 APPL AUTH=(ACQ,NVPACE),EAS=1
&lsvt22 APPL AUTH=(ACQ,NVPACE),EAS=1
&lsvt23 APPL AUTH=(ACQ,NVPACE),EAS=1
&lsvt24 APPL AUTH=(ACQ,NVPACE),EAS=1
&lsvt25 APPL AUTH=(ACQ,NVPACE),EAS=1
&lsvt26 APPL AUTH=(ACQ,NVPACE),EAS=1
&lsvt27 APPL AUTH=(ACQ,NVPACE),EAS=1
&lsvt28 APPL AUTH=(ACQ,NVPACE),EAS=1
&lsvt29 APPL AUTH=(ACQ,NVPACE),EAS=1
&lsvt30 APPL AUTH=(ACQ,NVPACE),EAS=1
&lsvt31 APPL AUTH=(ACQ,NVPACE),EAS=1
&lsvt32 APPL AUTH=(ACQ,NVPACE),EAS=1
&lsvt33 APPL AUTH=(ACQ,NVPACE),EAS=1
&lsvt34 APPL AUTH=(ACQ,NVPACE),EAS=1
&lsvt35 APPL AUTH=(ACQ,NVPACE),EAS=1
&lsvt36 APPL AUTH=(ACQ,NVPACE),EAS=1
&lsvt37 APPL AUTH=(ACQ,NVPACE),EAS=1
&lsvt38 APPL AUTH=(ACQ,NVPACE),EAS=1
&lsvt39 APPL AUTH=(ACQ,NVPACE),EAS=1
&lsvt40 APPL AUTH=(ACQ,NVPACE),EAS=1
```

Figure 5 (Part 2 of 3). APPL Statements in KLSVTLST

```
&lsvt41 APPL AUTH=(ACQ,NVPACE),EAS=1
&lsvt42 APPL AUTH=(ACQ,NVPACE),EAS=1
&lsvt43 APPL AUTH=(ACQ,NVPACE),EAS=1
&lsvt44 APPL AUTH=(ACQ,NVPACE),EAS=1
&lsvt45 APPL AUTH=(ACQ,NVPACE),EAS=1
&lsvt46 APPL AUTH=(ACQ,NVPACE),EAS=1
&lsvt47 APPL AUTH=(ACQ,NVPACE),EAS=1
&lsvt48 APPL AUTH=(ACQ,NVPACE),EAS=1
&lsvt49 APPL AUTH=(ACQ,NVPACE),EAS=1
&lsvt50 APPL AUTH=(ACQ,NVPACE),EAS=1
&lsvt51 APPL AUTH=(ACQ,NVPACE),EAS=1
&lsvt52 APPL AUTH=(ACQ,NVPACE),EAS=1
&lsvt53 APPL AUTH=(ACQ,NVPACE),EAS=1
&lsvt54 APPL AUTH=(ACQ,NVPACE),EAS=1
&lsvt55 APPL AUTH=(ACQ,NVPACE),EAS=1
&lsvt56 APPL AUTH=(ACQ,NVPACE),EAS=1
&lsvt57 APPL AUTH=(ACQ,NVPACE),EAS=1
&lsvt58 APPL AUTH=(ACQ,NVPACE),EAS=1
&lsvt59 APPL AUTH=(ACQ,NVPACE),EAS=1
&lsvt60 APPL AUTH=(ACQ,NVPACE),EAS=1
&lsvt61 APPL AUTH=(ACQ,NVPACE),EAS=1
&lsvt62 APPL AUTH=(ACQ,NVPACE),EAS=1
&lsvt63 APPL AUTH=(ACQ,NVPACE),EAS=1
&lsvt64 APPL AUTH=(ACQ,NVPACE),EAS=1
&lsvt65 APPL AUTH=(ACQ,NVPACE),EAS=1
&lsvt66 APPL AUTH=(ACQ,NVPACE),EAS=1
&lsvt67 APPL AUTH=(ACQ,NVPACE),EAS=1
&lsvt68 APPL AUTH=(ACQ,NVPACE),EAS=1
&lsvt69 APPL AUTH=(ACQ,NVPACE),EAS=1
&lsvt70 APPL AUTH=(ACQ,NVPACE),EAS=1
```

Figure 5 (Part 3 of 3). APPL Statements in KLSVTLST

Other APPL Parameters

When defining or changing virtual terminals in APPL statements, pay particular attention to the following parameters:

- PARSESS
- AUTH
- EAS
- MODETAB
- DLOGMOD
- SRBEXIT
- SESSLIM

PARSESS Parameter

Certain applications, such as the CT/Engine operator facility, support *parallel sessions*. A parallel session lets many users share a virtual terminal to communicate with the application. PARSESS=YES indicates that the application supports parallel sessions. PARSESS=NO is the default.

For more information on parallel sessions, see "Virtual Terminal Sharing" on page 46.

AUTH Parameter

The AUTH parameter determines whether this application program has the authority to use certain VTAM functions.

ACQ	Determines whether CT/Engine can issue OPNDST ACQUIRE and SIMLOGON.
NVPACE	Deactivates VTAM message traffic on sessions using this path. NVPACE is optional; it is used for performance purposes only.
PASS	Determines whether the application can issue CLSDST PASS.

EAS Parameter

The EAS parameter specifies the approximate number of concurrent sessions between the application and other logical units (LU-to-LU sessions). VTAM uses the specified value in a lookup scheme to find the representation of a session between the application program and another logical unit.

Specifying the proper EAS parameter value for virtual terminal ACBs can save a significant amount of common storage.

EAS=1 causes VTAM to chain all the application's sessions together and run that one chain for all traffic to the application. EAS=1 is *recommended* for all applications *except*

- virtual terminals that have more than 44 concurrent application sessions (with different applications)
- virtual terminals that support parallel sessions (for example, the CT/Engine operator facility)

The default is EAS=509 (for VTAM 3.2–3.4) or EAS=491 (for earlier releases of VTAM). Values greater than 44 and less than 494 allocate a half-page lookup table. VTAM looks up the network addresses in incoming traffic and searches the chain selected for the session block.

An EAS value greater than 494 allocates a full-page lookup table in ECSA. The performance gain takes effect only when hundreds of users access a single application.

Candle recommends that you set the EAS parameter according to the following guidelines:

- For virtual terminals that support parallel sessions, specify EAS=2000.
- For virtual terminals with more than 44 concurrent application sessions, follow the guidelines provided in the IBM manual *VTAM Customization*.
- For most other applications, specify EAS=1.

  To conserve storage, specify LIMIT=44 in the VSM DEFINE command. (See the *Operator's Guide*.)

For example:

```
&lsvt11 APPL  EAS=1,AUTH=(ACQ,NVPACE),ACBNAME=&lsvt11
```

MODETAB Parameter

In defining virtual terminals, you may also choose a logmode table to associate each logmode entry with a set of session parameters for the application. To do so, specify the MODETAB parameter in the virtual terminal APPL definition. If CT/Engine does not find an entry for MODETAB, it searches the IBM default logmode table (ISTINCLM).

See “Defining the Logmode Table Entry” on page 44. For information regarding the implementation of KLSINCLM, see the *Program Directory*.

DLOGMOD Parameter

The DLOGMOD parameter specifies the name of the default logmode table entry. (See “Defining the Logmode Table Entry” on page 44.)

Candle provides a default of D4B32782 (for model 2). This default is ignored upon successful logmode resolution.

For information about default logmodes for virtual printers, see “Virtual Printers” on page 57.

SRBEXIT Parameter

SRBEXIT=YES specifies that you want VTAM exit routines to run in SRB mode (supervisor state, key 0), rather than in TCB mode (problem state). If SRBEXIT=YES, the address space must be APF-authorized.

SESSLIM Parameter

The SESSLIM parameter allows a session limit for a VTAM application. If the SESSLIM parameter is specified for a virtual terminal used by a DEDICATE pool, SIMLOGON and multiple CLSDST PASS will not work. Therefore, you should not use the SESSLIM parameter for VTAM APPL statements that define virtual terminals in a DEDICATE pool.

For more information on virtual terminal pools, see “Virtual Terminal Pools” on page 41.

Chapter 3. Virtual Terminal Pools

A set of virtual terminals is called a *virtual terminal pool*. Each pool specifies the virtual terminals that make up the pool and describes the characteristics of the applications that will use the pool.

After you have defined virtual terminals to VTAM, as explained in “Virtual Terminals” on page 33, you can either select a predefined virtual terminal pool or you can use the VSM command to define a new virtual terminal pool or modify an existing pool. You can also select virtual terminal pools dynamically at session initiation time.

When you define an application session, you specify a virtual terminal pool that matches the target application's expectations and behavior. When a user starts a session with the application, CT/Engine selects a virtual terminal from the pool that you specified.

Use the VSM Command to define virtual terminal pools to CT/Engine. Use APPLDEF commands and session profiles to associate pools with applications. Be sure to specify a virtual terminal pool in a multisession or single session application definition. Do *not* specify a virtual terminal pool for CLSDST PASS application definitions.

Topics discussed in this chapter are

- selecting predefined pools
- defining a new virtual terminal pool
- defining the logmode table entry
- virtual terminal sharing
- TSO considerations
- CICS™ considerations
- how virtual terminals are selected from virtual terminal pools
- selecting a virtual terminal pool dynamically
- assigning specific virtual terminals within a pool
- defining virtual terminals dynamically
- deleting virtual terminal information
- displaying session information for a virtual terminal

Selecting Predefined Pools

CL/SUPERSESSION and CL/GATEWAY use the APPLDEF command to associate an application with the correct virtual terminal pool.

Table 4 on page 42 correlates application types with distributed virtual terminal pools.

Table 4. Default Virtual Terminal Pools

Type of Application	Pool
CT/Engine operator facility	VIRTPARS
TSO	TSOPOOL
TSO with VTAM 3.3 ESA and above	TSOESA
CICS with Autoinstall	VIRT3270
CICS without Autoinstall	&DEFPOOL
NetView® or NCCF	VIRTPASS
IMS	&DEFPOOL
IIN	IINPOOL
OMEGAMON®	VIRT3270
HCF	HCFPOOL
VM	VIRT3270
VM/VSCS	VIRT3270
CA-IDMS®/DC	VIRT3270
CA-ROSCOE®	VIRT3270
Most other applications	VIRT3270

Since the default pools are already part of the configuration, you need only specify them in the CL/GATEWAY APPLDEF command. For example, to define a TSO application named TSOA, you would use the pool named TSOPOOL. Your APPLDEF command would look like this:

```
APPLDEF TSOA                              -
        DEST(your network name for TSO)   -
        DESC('TSO SYSTEM A')              -
        MESSAGE('UP AT 2:00 P.M.')        -
        ORDER(1)                          -
        MULTSESS(YES)                     -
        POOL(TSOPOOL)
```

For detailed information on the APPLDEF command, see "Defining Applications" on page 88. Also, refer to the *OG/OS2* for more information.

Defining a New Virtual Terminal Pool

When you followed the instructions in the *Basic Configuration Guide*, you chose the proper virtual terminal pool for each application and increased the virtual terminal pool size, if you needed to do so. To customize further, you might want to define your own virtual terminal pools.

The VSM DEFINE command defines a pool of virtual terminal logical units (LUs) can be shared by products (such as CL/SUPERSESSION) that use CT/Engine. VSM DEFINE commands are normally included in the startup command list to ensure that all virtual terminal pools are available as soon as CT/Engine initialization is complete.

You can issue more than one VSM DEFINE command for the same pool. After you define a pool, subsequent VSM DEFINE commands add virtual terminals to that pool. This is useful if you want to expand the number of virtual terminals without stopping and starting the CT/Engine address space.

You can have one VSM DEFINE command for each virtual terminal. However, if your naming convention uses sequential numbering, you can use the THROUGH parameter to specify a series of virtual terminals.

Note: If CL/SUPERSESSION will have other Candle products as menu selections, refer to those product manuals for VSM requirements.

Figure 6 shows some of the default VSM commands contained in *&thilev.*TLSCMDS(KLS$VSMS).

```
*-------------------------------------------------------------------
*  STANDARD VSM POOLS FOR VIRTUAL TERMINAL SESSIONS.
*-------------------------------------------------------------------
*
VSM DEF VIRTPARS &lsvt00        LOGMODE(&DEFLMODE) DEFER PARALLEL
VSM DEF VIRTDED  &lsvt01 TH(10) LOGMODE(&DEFLMODE) DEFER DEDICATE
VSM DEF VIRTPASS &lsvt11 TH(70) LOGMODE(&DEFLMODE) DEFER PASS
VSM DEF VIRT3270 &lsvt11 TH(70) LOGMODE(&DEFLMODE) DEFER
VSM DEF MODEL2   &lsvt11 TH(20) LOGMODE(&DEFLMODE) DEFER
VSM DEF MODEL3   &lsvt21 TH(30) LOGMODE(&DEFLMODE) DEFER
VSM DEF MODEL4   &lsvt31 TH(40) LOGMODE(&DEFLMODE) DEFER
VSM DEF MODEL5   &lsvt41 TH(50) LOGMODE(&DEFLMODE) DEFER
VSM DEF MODEL9   &lsvt51 TH(60) LOGMODE(&DEFLMODE) DEFER
```

Figure 6. Default VSM Pools in &thilev.TLSCMDS(KLS$VSMS)

The first command in the KLS$VSMS member defines the pool VIRTPARS with the following parameters:

VIRTPARS	Specifies the pool to associate with a series of virtual terminals.
PARALLEL	Means that a virtual terminal in this pool can have more than one concurrent session with a single VTAM application.

The second VSM DEFINE command in Figure 6 includes the parameters:

VSM DEF	Denotes the Virtual Session Manager DEFINE command. It defines the virtual terminal pool used to establish virtual sessions.
VIRTDED	Specifies the pool to associate with a series of virtual terminals.
&lsvt00	Is the first APPLID or LU name in the VSM command that defines the pool.
TH(10)	Generates entries for terminals &lsvt01 through &lsvt10.
LOGMODE	Specifies the VTAM logmode table entry used to establish the virtual session.
DEFER	Means that the VTAM ACB OPEN for the virtual terminal ACB is deferred until a user activates a session.
DEDICATE	Specifies that a virtual terminal in session with another application cannot be shared with other pools.

The third and fourth VSM DEFINE commands include two additional parameters, **PASS** and **TIMEOUT**.

PASS	Specifies that sessions established through this VSM pool must be passed only once during the life of the session.
TIMEOUT	Specifies the maximum inactivity period allowed for virtual sessions that use this pool.

For a complete description of the VSM DEFINE command, see the *OG/OS2*.

Defining the Logmode Table Entry

To establish a virtual session, VTAM must find in a logmode table the logmode entry name specified during the logon request. VTAM searches first in the logmode table specified with the MODETAB parameter of the APPL statement that defines the virtual terminal. If the logmode entry name is not there, VTAM then searches in the IBM default logmode table (ISTINCLM). For more information about this table, see the IBM document called *VTAM Customization*.

CL/GATEWAY uses the LOGMODE parameter of the VSM DEFINE command to select a new logmode entry name for the virtual terminal pool.

It is important to remember that a virtual session between CT/Engine and the application does not have to abide by the same bind rules as a session between the physical device and the application. CT/Engine checks for the following session characteristics:

- SNA or non-SNA
- queriable or nonqueriable
- screen size

The logmode used by CL/GATEWAY applies to the virtual terminal, not the physical terminal. CT/Engine does not support applications that attempt to use more than one explicit partition. The virtual terminal can be SNA or non-SNA, regardless of the session protocol for the physical terminal. The virtual terminal screen size can have different characteristics than the physical terminal. The virtual terminal supports the 3270 QUERY order on queriable sessions, even if the physical terminal does not.

Applications that Set Their Own Session Parameters

When an application session begins, a logmode entry from the terminal suggests session parameters. IMS and CICS (without Autoinstall) override the session parameters suggested by the terminal, and use instead the parameters found in their own tables.

For IMS, the logmode must be non-SNA if the IMS definitions are non-SNA, and SNA if SLU2 TERMTYPES are used. This requirement is independent of the physical session and logmode.

CICS without Autoinstall ignores the logmode name in the CINIT (terminal logon request) when the virtual terminal device is defined in the TCT. See “CICS Considerations” on page 48.

Although you cannot use logmodes to control session parameters with these applications, you can use pools for this purpose. For these applications, therefore, you should set up a separate pool for each terminal type (model 2, 3, 4, 5, or 9) whose characteristics must carry to the application.

Selecting a Logmode Entry for Extended Functions

Virtual terminal sessions with extended datastream functions, APL functions, or both must use a logmode entry for a queriable LU0 or LU2 device. *&thilev*.TLSSAMP(KLSINCLM) contains a sample logmode table for these sessions. If you intend to use extended datastream support, and if compatible logmodes are not available in the logmode table that you plan to use for virtual terminals, you may need this supplementary table.

Some applications require or perform better with a specific set of session characteristics. Use the administrator functions to customize your logmode table for those applications' pools. See the *Basic Configuration Guide*.

Selecting a Logmode Entry Dynamically

When a user accesses an application, CL/GATEWAY (with or without CL/SUPERSESSION) invokes the dialog KLSLMOD. The KLSLMOD dialog looks for the logmode table entry in the online logmode table. Otherwise, the product uses default logmodes.

To set the default logmode entry name for virtual sessions to the logmode entry name used for the physical terminal, perform the following steps:

1. Make sure that the CT/Engine virtual devices are referencing the proper logmode table.

   If you are using the default logmode table, skip to step 2. Otherwise, check that the MODETAB parameter is referencing the correct table on all the virtual terminal APPL definitions in SYS1.VTAMLST(*newname*). For example:

   ```
   KLST0001   APPL   AUTH=(ACQ,NVPACE),MODETAB=KLSINCLM
   KLST0002   APPL   AUTH=(ACQ,NVPACE),MODETAB=KLSINCLM
   ```

2. From the Administrator Menu, select **`Update Logmode Table`**. Make sure that the value for the DEFLMODE field of the DEFAULT entry is D4B32782.

   For detailed information on using the administrator functions to create and modify the logmode table, see the *Basic Configuration Guide*.

Note: The logmode table created through the administrator functions overrides any customization you do in dialog KLSLMOD. Therefore, you should customize your logmode table through the administrator functions, not through KLSLMOD. Do not modify dialog KLSLMOD.

Unspecified Screen Size Bind

CT/Engine supports the unspecified screen size bind as implemented by 3x74 control units for 3270 terminals. This support allows all terminals to use a common logmode. Use of a common logmode simplifies the task of changing, adding, and moving terminals in the network, and eliminates the need for network users to remember logmode names.

When byte 24 of the terminal bind image contains x'03' (that is, the unspecified screen size indication), the default screen size for the terminal is 24 lines by 80 columns. The implicit partitions query reply from the terminal is examined to obtain the alternate screen size for the terminal. If an implicit partitions query reply cannot be obtained from the terminal, both the default and alternate screen sizes are set at 24 lines by 80 columns (i.e., 3270 Model 2, 24x80).

CL/GATEWAY and CL/SUPERSESSION offer support for the unspecified screen size bind on virtual sessions with host applications. When the host application does not support unspecified screen size binds, dialog logic (aided by logmode customization through online administrator functions) can determine an appropriate logmode name for those virtual sessions. In this way you can convert the network to use a common unspecified screen size bind image and compensate for host applications that do not currently support unspecified screen size binds.

Explicit Partition Operation

CL/GATEWAY and CL/SUPERSESSION support host applications that use the explicit partition capability of certain 3270-family terminals. This capability, which is used by NetView and GDDM™, is available in the following commonly used terminals:

- 3179G
- 3180
- 3192D
- 3192G
- 3290

Support is currently limited to a single explicit partition with identifier zero.

Virtual Terminal Sharing

When you introduce virtual terminals to your network, you introduce additional VTAM resources, because virtual terminals use CSA and other resources. With the proper virtual terminal sharing techniques, you can cut down on VTAM resource requirements for your virtual terminals.

There are two types of virtual terminal sharing:

- *ACB sharing*, in which an OPEN-ACB can have OPEN-DESTs to sessions with more than one VTAM application
- *parallel sessions*, in which an OPEN-ACB can have more than one OPEN-DEST to the same VTAM application

Virtual devices can be shared across more than one virtual terminal pool and can also be shared among VTAM applications. For example, the same virtual terminal can be used for a virtual session with TSO, IMS, CICS, and NetView, for either a single user session or a combination of user sessions. However, the virtual terminal cannot be in session with the same application more than once, unless both the virtual terminal and the application can support parallel sessions.

Parallel sessions can be established only if

- the VTAM application APPL can support parallel sessions and has coded in SYS1.VTAMLST(*newname*)
- the virtual device has PARSESS=YES coded in SYS1.VTAMLST(*newname*)
- the virtual terminal pool definition (the VSM DEFINE command) specifies the PARALLEL parameter

CT/Engine, OMEGAMON, and TSO can support parallel sessions.

Unless you use parallel sessions, each application requires as many virtual devices as the maximum number of concurrent users.

Applications (such as IIN™) that use *more than one CLSDST PASS* to pass sessions inhibit virtual terminal sharing. The virtual terminal treats a passed session as a session termination followed by a session initiation, and receives no information to correlate the two sessions. If two such session passes are *in transit* at the same time, the virtual terminal cannot determine which of them was completed first.

The virtual terminal cannot control session passing. To avoid crossed sessions, CT/Engine requires that you specify in advance which sessions will be passed (CLSDST PASS) more than once. Terminals that support such a session are not selected to support another, and should be assigned to a pool with the DEDICATE parameter.

Applications that pass a terminal (CLSDST PASS) *exactly once* and do so quickly (for example, NetView) only temporarily inhibit sharing, while the pass is occurring. Once a session has been passed, it is not passed again, and the virtual terminal can support another such session. This type of application should be assigned to a pool with the PASS parameter.

If a pool does not contain enough virtual terminals for your needs, you may see this message on your screen:

```
CS031 RESOURCES UNAVAILABLE
```

In this case, be sure to increase your pool size. Otherwise, your users may not be able to establish the sessions they need.

TSO Considerations

TSO's RECONNECT processing requires more than one CLSDST PASS. To accommodate this requirement, the virtual terminal pool definition must include the DEDICATE parameter, which does not allow parallel sessions or ACB-sharing.

Therefore, if your users will have SINGLE session (rather than MULTI session) access to TSO, you may want to use the DEDICATE parameter of the VSM DEFINE command when you define virtual terminal pools for TSO. (See the *OG/OS2*.)

However, because CL/SUPERSESSION establishes physical sessions without breaking virtual sessions, defining a DEDICATE pool for TSO as a multisession application is unnecessary and wastes resources.

To use graphics with TSO, make sure that the physical terminal is using the proper logmode and can support extended data streams.

TSO hardware partitioning of the 3290 is not supported. TSO uses more than one explicit partition with non-zero identifiers for the 3290 devices.

CICS Considerations

Your customization needs for CICS virtual terminals depend on whether you are using CICS with or without Autoinstall.

CICS without Autoinstall

CICS uses a terminal control table (TCT) to describe virtual terminal characteristics. It does not use the VTAM description of the terminal. In the TCT you must define all terminals, both physical and virtual, that will be in session with CICS. *&thilev.*TLSSAMP(KLSCTCT) gives sample TCT entries.

This example shows a VSM pool definition that you can use to define virtual terminals for a CICS region.

```
VSM DEF CICSMOD3  KLST0001  TH(50)  LOGMODE(&DEFLMODE) DEFER
```

You must specify separate TCT entries for each of the KLST0001-KLST0050 virtual terminals defined in this VSM command. You should specify the following parameters:

- NETNAME=KLST0001 to define the KLST0001 virtual terminal.
- The parameters that determine terminal model. For example, a model 3 requires the DEFSCRN=(32,80) or ALTSCRN=(32,80) parameter, depending on how you specify the screen size in your CICS region's TCT.
- TRMTYPE to specify whether the terminal is SNA or non-SNA.

- FEATURE to specify certain hardware features that different terminal models might have. For example, for model 3s, you might include *color, highlight,* or *ps*.

```
KLST0050  DFHTCT  TYPE=TERMINAL,                             X
          TRMIDNT=M050,                                                X
          TRMTYPE=LUTYPE2,                                             X
          ACCMETH=VTAM,                                                X
          ALTSCRN=(32,80),                                             X
          ALTSFX=3,                                                    X
          BRACKET=YES,                                                 X
          BUFFER=1536,                                                 X
          RUSIZE=2048,                                                 X
          CHNASSY=YES,                                                 X
          DEFSCRN=(24,80),                                             X
          ERRATT=(LASTLINE,INTENSIFY),                                 X
          FEATURE=(DCKYBD,UCTRAN,3270E,AUDALARM,SELCTPEN,              X
          COLOR,HILIGHT,PS),                                           X
          GMMSG=YES,                                                   X

          NETNAME=&lsvt50,                                   X

          RELREQ=(NO,YES),                                             X
          TIOAL=(4000),                                                X
          TRMMODL=2,                                                   X
          TRMPRTY=0,                                                   X
          TRMSTAT=(TRANSCEIVE)X
          LOGMODE=0
```

Figure 7. TCT Entry for the First Terminal in the CICSMOD3 VSM Pool (SNA)

Figure 7 and Figure 8 each show a TCT entry in a production CICS system. The entry in Figure 7 sets up an SNA terminal; the entry in Figure 8 sets up a non-SNA terminal. Both entries work in conjunction with the following CICSMOD3 pool definition:

```
VSM DEF CICSMOD3  KLST0001  TH(50)  LOGMODE(D4A32783) DEFER
```

Note: This is only an example; your TCT definitions will be different.

```
KLST0050 DFHTCT TYPE=TERMINAL,                               X
               ACCMETH=VTAM,                                           X
               NETNAME=KLST0050                                        X
               RELREQ=(NO,YES),                                        X
               GMMSG=YES,                                              X
               FEATURE=(DCKYBD,UCTRAN,3270E,AUDALARM,SELCTPEN,         X
               COLOR,HILIGHT,PS),                                      X
               ERRATT=(LASTLINE,INTENSIFY),                            X
               BRACKET=YES,                                            X
               RUSIZE=2048,                                            X
               TRMIDNT=M050,                                           X
               TRMMODL=2,                                              X
               TRMPRTY=0,                                              X
               TRMSTAT=(TRANSCEIVE),                                   X
               TRMTYPE=L3277,                                          X
               DEFSCRN=(24,80),                                        X
               ALTSCRN=(32,80),                                        X
               TIOAL=3000,                                             X
               ALTSFX=3X
               LOGMODE=0

(Non-SNA)
```

Figure 8. TCT Entry for the First Terminal in the CICSMOD3 VSM Pool

CICS with Autoinstall

When defining virtual terminals for use by a CICS system with automatic installation of terminals (Autoinstall), you do not need to specify discreet pools of virtual terminals for use by particular model types. Nor do you need to specify the NODE parameter on the VSM DEFINE.

Instead, CICS uses automatic installation models defined in RDO with CEDA, and a user program to map the terminal name to the netname. IBM supplies an example user program (called DFHZATDX), coded in Assembler. If automatic installation is active, the user program receives control at installation when a VTAM logon request is received from a resource whose NETNAME is not in the TCT. The user program is passed a parameter list of five contiguous fullwords:

1. Function field. Byte 1 indicates the request type. This is `X'F0'` for install. The remaining three bytes are reserved.
2. Pointer to identifier field. The identifier field consists of a 2-byte length field, followed by the NETNAME of the resource requesting LOGON, for example, a CL/SUPERSESSION virtual terninal.
3. A pointer to an array of names of eligible automatic installation models. The array is preceded by a 2-byte field describing the number of the 8-byte name element in the array.
4. Pointer to a 21-byte field to return information to the calling program.
   - AUTINSTNAME - 8 bytes
   - TERMINAL - 4 bytes
   - PRINTER - 4 bytes
   - ALTPRINTER - 4 bytes
   - Status byte - 1 byte

   To return information, the user program sets the required information in the return information field. The status byte must be set to binary zeros.
5. Pointer to the VTAM CINIT request unit. The data is preceded by a 2-byte length field indicating the length of the CINIT request unit.

See the IBM manual *CICS Customization Guide* for more information on the autoinstall process.

The CINIT request unit can be parsed to locate any userdata passed with the request. Candle supplies an example user program to locate userdata supplied with a logon request in *&thilev*.TLSSAMP(KLSZATDX).

To pass the physical terminal to the autoinstall user program, you must define the CICS application with USERDATA=`'&SYSTERM'` on the APPLDEF command.

How Virtual Terminals Are Selected from Virtual Terminal Pools

CL/SUPERSESSION and CL/GATEWAY assign virtual terminals in last-in-first-out (LIFO) sequence. You can use the NODE parameter of the VSM DEFINE command to change the virtual terminal assignment sequence. See “Assigning Specific Virtual Terminals within a Pool” on page 52.

Selecting a Virtual Terminal Pool Dynamically

CL/SUPERSESSION and CL/GATEWAY let you select a virtual terminal pool dynamically.

When you specify POOL=*vsmpool* in an APPLDEF command, *vsmpool* identifies a specific virtual terminal pool defined by a VSM DEF command. When you specify POOL='&DEFPOOL', the virtual terminal pool is assigned dynamically from the DEFPOOL variable, which is set through the administrator functions. (See the *Basic Configuration Guide*.)

Note: If you do not specify the POOL parameter on the APPLDEF command, the gateway passes terminal control directly to the destination application and breaks the gateway session with the terminal (VTAM CLSDST PASS).

For detailed information about the APPLDEF command, see “Defining Applications” on page 88. Also, refer to the *OG/OS2* for more information.

Example: CICS Dynamic Pool Selection (CL/GATEWAY with CL/SUPERSESSION)

As shown in this example, CICS regions that use a TCT to define the characteristics of eligible physical terminals might require you to select the terminal pool dynamically.

Candle provides sample gateway configuration members, APPLDEF commands, VSM pools, and dialogs to handle dynamic pool selection for most terminal types.

Here is an APPLDEF command for a CICS application. This definition is *&rhilev*.RLSCMDS(KLSCAPLS).

```
APPLDEF CICSSS                             -
        DEST=CICS1                         -
        DESC='CICS (PASSTHRU/MULTSESS)'    -
        POOL='&DEFPOOL'                    -
        MULTSESS=YES                       -
        GROUP=200
```

In this example, the value of the APPLDEF POOL parameter defaults to the value of the variable &DEFPOOL. The variable &DEFPOOL selects a virtual terminal pool dynamically, based on the terminal model. When a user signs onto the gateway, the value of &DEFPOOL is set by the logmode table customized and maintained through the administrator functions.

Assigning Specific Virtual Terminals within a Pool

The NODE parameter of the VSM DEF command controls terminal allocation in a specific virtual terminal pool. Once you select a pool, the NODE parameter determines which virtual terminal is allocated from that pool.

The NODE parameter can specify

- any valid string expression or literal string
- a variable to be set by a dialog during session initiation

  This is useful if you want to select the virtual terminal by either user ID or physical terminal ID. The variable used as the NODE value must be declared as a shared variable in *&thilev.*TLSPNLS(KLSSDCL). If this is not done, when a new window is created the NODE variable will be nulls and NODE will not be resolved.

Note: For a string expression in a VSM command, use &SYSEDIT, not &SUBSTR. (&SUBSTR does not work with NODE.) The format for &SYSEDIT is

```
&SYSEDIT((offset,length)string)
```

If you do not specify the NODE parameter, virtual terminals are assigned in LIFO sequence.

Mapping Virtual Terminals

The following examples show how to use more than one VSM DEF command to define a single virtual terminal pool. These virtual terminal mapping techniques use the NODE parameter of the VSM DEF command. The variable used as the NODE value must be declared as a shared variable.

Example 1: Changing the First Character of the Physical Terminal Name

This example changes the first character of the physical terminal network name to V, where the physical terminal names are KLST0002, KLST0003, and so on.

```
VSM DEF VIRTPOOL VLST0002 LOGMODE=&DEFLMODE DEFER  -
                 NODE='V&SYSEDIT((1,7)&SYSTERM)'
VSM DEF VIRTPOOL VLST0003 DEFER
VSM DEF VIRTPOOL VLST0004 DEFER
VSM DEF  . . .     . . .  . . .
```

Example 2: Prefixing the Physical Terminal Name

This example prefixes the physical terminal network name X, where the physical terminal names are KLST001, KLST002, and so on.

```
VSM DEF VIRTPOL2 XLST001 LOGMODE=&DEFLMODE DEFER  -
                 NODE='X&SYSTERM'
VSM DEF VIRTPOL2 XLST002 DEFER
VSM DEF VIRTPOL2 XLST003 DEFER
VSM DEF  . . .    . . .   . . .
```

Example 3: Suffixing the Physical Terminal Name

This example suffixes the physical terminal network name Z. The Z is either used as the last character of a seven-character name, or substituted for the last character of an eight-character name. For this example, the physical terminal names are KLST001, KLST002, and so on.

Note: This example assumes that all physical terminal network names are either seven or eight characters long.

```
VSM DEF VIRTPOL3 KLST001Z LOGMODE=&DEFLMODE DEFER  -
                 NODE='&SYSEDIT((0,7)&SYSTERM)Z'
VSM DEF VIRTPOL3 KLST002Z DEFER
VSM DEF VIRTPOL3 KLST003Z DEFER
VSM DEF  . . .    . . .   . . .
```

Example 4: Using a Complex String Expression

This example uses a complex string expression.

```
VSM DEF VIRTPASS KLST60   TH(9) LOGMODE(&DEFLMODE) PASS -
                          NODE='KLST&SYSEDIT((1,3)&SYSTERM)'
VSM DEF VIRTPASS KLST60A
VSM DEF VIRTPASS KLST60B
VSM DEF VIRTPASS KLST60C
VSM DEF VIRTPASS KLST60D
VSM DEF VIRTPASS KLST60E
VSM DEF VIRTPASS KLST60F
```

The virtual terminal name is derived by appending the second through fourth characters of the physical terminal ID to the character string KLST. The &SYSEDIT function extracts the second through fourth characters from the physical terminal name, &SYSTERM.

Because the THROUGH parameter cannot be used to define non-numeric virtual terminal suffixes, several VSM commands are used to define this pool.

You can define this pool in association with a series of non-SNA physical terminals whose names contain channel unit addresses. For example, a group of VTAM physical terminals with names T600 through T60F can be mapped to virtual terminals KLST600 through KLST60F by this method.

Example 5: Using the ? Wildcard Character

This example uses the question mark (?) wildcard character to assign users from two locations to virtual terminals that reflect their respective locations.

The first 3 characters of the physical terminals for the two locations are APT and BPT.

```
VSM DEF CICSPOOL APT000 TH(030) LOGMODE=&DEFLMODE DEFER -
                  NODE='&SYSEDIT((0,3)&SYSTERM).???'
VSM DEF CICSPOOL BPT000 TH(030) LOGMODE=&DEFLMODE DEFER
```

Note: The NODE parameter is propagated to all terminals in a pool, so it is required on only one VSM DEF command.

Defining Virtual Terminals Dynamically

If you have a large number of virtual terminals to allocate, using VSM DEF commands to create and maintain the necessary pools might involve a great deal of work. To simplify the task, you can create virtual terminals dynamically, as users sign on, by using a physical-to-virtual terminal one-to-one mapping technique. Each virtual terminal name is derived from a specific physical terminal ID or user ID.

This example dynamically adds a virtual terminal to the pool as the user logs on. The advantage of this technique is that you don't have to issue VSM commands to maintain virtual terminal pools, and you save on ACB storage. Each virtual terminal name in this example consists of the last four characters of the physical terminal name. For example, physical terminal ID C12AZ0A3 maps to virtual terminal ID Z0A3.

1. Create *dummy* virtual terminal pools in *&rhilev*.RLSCMDS member KLS$VSMS. For example,

```
VSM DEF PASSPOOL DUMMY LOGMODE(&DEFLMODE) DEFER PASS -
                       NODE='&VIRTTERM'
VSM DEF NOPASS DUMMY LOGMODE(&DEFLMODE) DEFER        -
                       NODE='&VIRTTERM'
```

2. Create a dialog in your *&rhilev*.RLSPNLS with the following dialog code:

```
)OPTIONS LEVEL (1)
)INIT
SET VIRTTERM '&SUBSTR('&SYSTERM',4,4)'
COMMAND('AS ''*SYSVLG*'' VSM DEF PASSPOOL &VIRTTERM DEFER')
COMMAND('AS ''*SYSVLG*'' VSM DEF NOPASS &VIRTTERM DEFER')
```

In this code, AS `'*SYSVLG*'` issues the commands on behalf of the VIEWLOG operator. The command replies are then recorded in the VIEWLOG dataset.

This dialog needs to be specified as a profile initial dialog so that it executes during a user's initialization after entry validation.

Note: This technique still requires that each virtual terminal is defined in SYS1.VTAMLST(*newname*) and that VIEWLOG is enabled.

Optional: To close a virtual terminal that has been dynamically allocated, review example codes in CLSEACB, CLSEACB1, CLSEACB2, and CLSEACB3 in *&rhilev*.RLSSAMP. The examples explain how to close ACBs that you dynamically allocated and prevent the virtual terminals from taking up extra storage.

Deleting or Redefining a Virtual Terminal Pool Definition

To delete a virtual terminal pool, use the VSM DELETE command. VSM DELETE deactivates all virtual sessions using virtual terminals in the specified pool, and closes the VTAM ACBs.

To redefine an existing pool, issue the VSM DELETE command, and then reissue a VSM DEFINE command to define the pool with new attributes.

Important

Use the VSM DELETE command with caution. If you delete a pool that contains virtual terminals defined in more than one pool, the virtual terminals are deleted from every pool in which they are defined. Furthermore, when you issue the VSM DELETE command, you disable all virtual terminals in the deleted pool, including those currently in use.

Use the CLOSE command, not the VSM DELETE command, to close an individual terminal. If you CLOSE an individual terminal in a specific pool, the terminal is deleted from *every* pool in which it is defined. If you CLOSE all the virtual terminals in a pool, the pool is deleted.

Do not delete the default pool $DEFAULT. If $DEFAULT is deleted or defined incorrectly, the results will be unpredictable.

For detailed information on CLOSE and VSM DELETE, see the *OG/OS2*.

Displaying Virtual Terminal Information

The VSM DISPLAY command provides detailed session information on the application or virtual terminal LU specified in the command. The information concerns

- active virtual sessions with a specific virtual terminal
- active virtual sessions with a specific VTAM application

The VSM LIST command displays statistical information about a virtual terminal pool. This information includes

- the identifiers of all pools, or of the specified pool only
- pool parameters specified in pool definitions
- the current number of each of the following:
  - Active sessions for virtual terminals in the pool. Because of virtual terminal sharing, this number could be significantly higher than the number of open sessions.
  - Available virtual terminals. Virtual terminals are considered available until they reach their limit of active sessions (specified with the LIMIT parameter of the VSM DEFINE command).
  - Open sessions (that is, ACBs opened by VTAM). If you specify DEFER and do not specify PARALLEL in the VSM DEFINE command, the number of open sessions provides the high-water mark for the number of active sessions with any one application.
  - Deferred sessions. The total number of open and deferred sessions should equal the number of virtual terminals defined for the pool.
  - Sessions allowed for each virtual terminal in the pool, as specified by the LIMIT parameter of the VSM DEFINE command.

For detailed information on the VSM DISPLAY and VSM LIST commands, see the *OG/OS2*.

Chapter 4. Virtual Printers

A virtual printer is an ACB associated with a network printer. Virtual printers let various applications share a physical printer.

For example, if you have one physical printer used by both IMS and CICS, you can create virtual printers that share the physical printer, with or without a release request (RELREQ) protocol. You can then route all printing to the virtual printer, which manages the physical printer and prevents conflicts among applications.

You can also implement a virtual printer to direct printing to SYSOUT. The advantage of using SYSOUT is that either JES or MVS handles routing to the appropriate device.

Here are some of the benefits of virtual printers:

- Virtual printers reduce resource consumption and increase device availability by eliminating the necessity to initiate and terminate a session with the physical printer for each print request.
- You can replace the physical printer definitions in application terminal control tables with static virtual printer definitions, which can then be associated dynamically with specific physical printers.
- Virtual printers permit a constant flow of print streams from many applications in first-in-first-out (FIFO) order.

Virtual Printer Sharing

Any number of CT/Engine and non-CT/Engine applications can share a virtual printer. A user can share a virtual printer with any other user, regardless of the application in use. Any application that requires access to a physical printer can gain access at any time.

CT/Engine manages the flow of data to the printer by enforcing the SNA bracket protocol. The SNA bracket protocol governs the attachment to the named physical printer. When several virtual printer sessions wish to begin a bracket, VPRINTER resolves their requests in FIFO sequence. Contention losers are sent an exception response sense data code of x'08020000' (Intervention Required). When a virtual session that has previously been a contention loser moves to the top of the pending bracket queue, an LUSTAT code x'00010000' is sent to notify the application that the physical printer is now available.

Using Virtual Printers

To implement a virtual printer, you must perform the following steps:

1. Define the virtual printer as an ACB in a major node in SYS1.VTAMLST(newname). Use the ACQ parameter to give the ACB acquire authority.

   CT/Engine passes all data to the physical printer unchanged. For this reason, the device characteristics of the virtual and physical printers must be identical. You might need to specify the DLOGMOD parameter on the virtual printer APPL statement in SYS1.VTAMLST(newname) to ensure proper coordination with the physical printer's characteristics. Candle recommends the logmode DSC2K for SNA printers and S3270 for non-SNA printers. For example,

   ```
   ACB16 APPL AUTH=(ACQ,NVPACE), ACBNAME=ACB16, PARSESS=YES,
   DLOGMOD=DSC2K
   ```

   or

   ```
   ACB16 APPL AUTH=(ACQ,NVPACE), ACBNAME=ACB16, PARSESS=YES,
   DLOGMOD=S3270
   ```

2. Issue the VPRINTER command, either from the startup command list, *&rhilev*.RLSCMDS(KLSSTART), or as an operator command. The VPRINTER command opens the ACB and associates the virtual printer with the physical printer. For example,

   ```
   VPRINTER ACB16 SYSOUT CLASS=S DEST=RMT5
   ```

   For detailed information on the VPRINTER command, see the *Operator's Guide*.

Now your users simply specify a virtual printer name instead of a physical printer name when they print. Or you can let the NAM database or the application select the virtual printer, making the selection transparent to the user. See “Network Access Manager” on page 147.

Once you issue the VPRINTER command, it opens the ACB. Any of the following actions closes the virtual printer ACB:

- issuing the CLOSE command
- shutting down CT/Engine
- issuing the VTAM VARY INACT command

If you are going to use virtual printers frequently, you will probably want to add the VPRINTER commands as the last few lines in your startup member.

Using VPRINTER when Printing Session Screens

If you will be printing session screens while using the VPRINTER function, you must be certain that the virtual printer ACB and the dialog ACB that is using the virtual printer (for example, the CL/SUPERSESSION ACB) are both defined in VTAMLST APPLs as parallel-session-capable (PARSESS=YES on the APPL statement). For example,

```
ACB16 APPL AUTH=(ACQ,NVPACE), ACBNAME=ACB16, PARSESS=YES, DLOGMOD=DSC2K
```

When VPRINTER is waiting for the physical printer, many active VSSPRINT sessions from the dialog ACB to the virtual printer ACB must be sustained. If the two applications are not parallel-session-capable, the first VSSPRINT will get a session and any subsequent VSSPRINT session requests will fail with RC20.

The distributed **@p** trigger can also be used to route session screens to VTAM printers. See the *User's Guide* for information on invoking this trigger.

For more information on the VSSPRINT statement, see the *Dialog Language Reference Manual*. For more information on the VPRINTER command, see the *Operator's Guide*.

Using Virtual Printers in Port Contention Environments

Port contention causes the network name (LU name) assigned to a physical terminal or printer to be unpredictable. However, many applications depend on the terminal network names both for security and for acquiring printers.

Port contention can be caused by any of the following:

- Value added networks, such as IIN or TYMNET
- Protocol converters, such as 3708s
- Local area networks (LANs)
- X.25

Although virtual terminal pooling does not cause physical port contention, it does create the same types of problems for applications that depend on the physical terminal name.

The following examples provide solutions to IIN port contention problems for CL/SUPERSESSION and CL/GATEWAY users.

Note: These examples may require additional customization for particular customer needs or environments.

CL/SUPERSESSION Example

1. Define all necessary virtual terminals as APPLs to VTAM, if you have not already done so.
2. Create dialogs IINPRT and IINPRTR in *&rhilev*.RLSPNLS, as shown in Figure 9 and Figure 10 on page 61.

   IINPRT is a profile initial dialog that calls IINPRTR to determine the IIN printer name.

```
)comment
Name: IINPRT

 To be used as a profile initial dialog.

 This dialog starts the appropiate VPRINTER based upon
 the user's ID.

)option level(1)

)declare
vspprt  scope(session)
viguser scope(session)
iinuid  scope(local)
l       scope(local)
o       scope(local)
vprt    scope(local)

)init
If '&substr(&systerm,0,3)' ne 'IBM'  /* is it an IIN user ? */
  return

set iinuid &viguser
)prologue

set l '&length(&iinuid)'     /* actual length of the USER ID */
set o &l-1           /* set the offset to the last char */
if &vspprt            /* Get the users default VPRINTER name */
  set vprt &vspprt /* from the PROFILE setting */
 else                                /* OR */
  set vprt '&substr(&iinuid,0,&l)P'   /* Add a P to the USERID */

COMMAND('as ''*SYSVLG*'' CLOSE &vprt')
wait(1)             /* let it close */

dialog IINPRTR    /* establish the actual IIN printer name */
If &sysrc
 do
  COMMAND('as ''*SYSVLG*'' VPRINTER &vprt &sysrc')
  return
 end

dialog klsmsgbl 'ERROR,R,P,The &vprt was not started'
```

Figure 9. CL/SUPERSESSION IINPRT Dialog

```
)comment
Name: IINPRTR   (called by IINPRT)

 This dialog determines the IIN printer name assigned
 based upon the name assigned for the IIN terminal.

)option level(1)

)declare
iintrm scope(local)
iinprt scope(local)
l      scope(local)
o      scope(local)
lc     scope(local)
ac     scope(local)

)init
set iintrm &systerm /* SYSTERM always has length of 8 */
)prologue

set l '&length(&iintrm)' /* actual length of the terminal ID */
set o &l-2       /* set the offset to the 2nd to last char */

if '&substr(&iintrm,&o,1)' eq 'J' /* is the 7th char a J ? */
  do
    set prt1 '&substr(&iintrm,0,&o)K' /* then the printer has a K */
    set prt2 '&substr(&iintrm,&l-1,1)'
    set iinprt '&prt1&prt2'
    return &iinprt
  end
else                                   /* if NOT .... */
 do
   set o &o+1  /* reset offset to the last char */
   set lc '1'           /* Starting point */
   set ac 'A'
 end

DO
 if '&substr(&iintrm,&o,1)' eq &lc /* Check the last char */
  do
   set iinprt '&substr(&iintrm,0,&o)&ac' /* set 1 thru 9 */
   Return &iinprt                         /*   to      */
  end                                         /* A thru I */
 set lc &lc+1
 set ac (eval '&&sysincr(&ac)')
UNTIL &lc gt 9

LOG('PRT = &iinprt lc=&lc ac=&ac l=&l o=&o TRM=&iintrm')   /* log the values  */
exit                  /* not an IIN terminal  ??  */
```

Figure 10. CL/SUPERSESSION IINPRTR Dialog

CL/GATEWAY-Only Example

In a CL/GATEWAY only connection through IIN the following procedure can be used.

Specify the PRTPOOL operand and PRINTER=REQUIRED on the APPLDEF as shown in the following example.

1. Define all necessary virtual terminals as APPLs to VTAM, if you have not already done so.
2. Add the following to *&rhilev*.RLSCMDS(KLGCAPLS):

   ```
   APPLDEF CICS dest=applid ... -
            PRTPOOL='*' PRINTER=REQUIRED
   ```

3. In the *&rhilev*.RLSPARM GATEWAY configuration, specify the following:

   ```
   PRTNODE  REQUIRED
            PANEL(IINPRTR)
   PRTPOOL  REQUIRED
            DEFAULT(VPRTP)
   ```

4. Add the following to *&rhilev*.RLSCMDS(KLS$VSMS):

   ```
   VSM DEF VPRTP VPRT00 TH(5) DEFER    -
            Logmode(&deflmode)
   ```

5. Create the IINPRTR dialog in *&rhilev*.RLSPNLS as shown in Figure 11 on page 63.

   A session is automatically established between the gateway elements &VIGPRTPL (PRTPOOL) and &VIGPNODE (PRTNODE).

```
)comment
Name: IINPRTR

 This dialog determines the IIN PRINTER name assigned
 based upon the name assigned for the IIN TERMINAL.

)option level(1)

)declare
iintrm scope(local)
iinprt scope(local)
l      scope(local)
o      scope(local)
lc     scope(local)
ac     scope(local)

)init
set iintrm &systerm /* SYSTERM always has length of 8 */

)prologue

set l '&length(&iintrm)' /* actual length of the terminal ID */
set o &l-2       /* set the offset to the 2nd to last char */

if '&substr(&iintrm,&o,1)' eq 'J' /* is the 7th char a J ? */
  do
    set prt1 '&substr(&iintrm,0,&o)K' /* then the printer has a K */
    set prt2 '&substr(&iintrm,&l-1,1)'
    set iinprt '&prt1&prt2'
    set vigpnode '&iinprt'
    return
  end
else                                   /* if NOT .... */
 do
   set o &o+1  /* reset offset to the last char */
   set lc '1'           /* Starting point */
   set ac 'A'
 end

DO
 if '&substr(&iintrm,&o,1)' eq &lc /* Check the last char */
  do
   set iinprt '&substr(&iintrm,0,&o)&ac' /* set 1 thru 9 */
   set vigpnode '&iinprt'
   Return                                      /*   to      */
  end                                          /* A thru I */
 set lc &lc+1
 set ac (eval '&&sysincr(&ac)')
UNTIL &lc gt 9

LOG('PRT = &iinprt lc=&lc ac=&ac l=&l o=&o TRM=&iintrm')   /* log the values  */
set vigpnode 'DEFPRINT'   /* to prevent element resolution loop */

exit                  /* not an IIN terminal  ??  */
```

Figure 11. CL/GATEWAY IINPRTR Dialog

Chapter 5. CL/SUPERSESSION and CL/GATEWAY Customization

This chapter explains the gateway configuration and shows how to customize CL/SUPERSESSION and CL/GATEWAY. You will learn how to

- set CL/GATEWAY initialization parameters
- define a network entry point
- specify a gateway data definition
- customize messages
- define applications
- create authorized application lists
- customize the network broadcast facility
- customize the news facility

Setting CL/GATEWAY Initialization Parameters

During CL/GATEWAY initialization, *&rhilev*.RLSPARM(KLGINGWY) provides default values for certain global CL/GATEWAY parameters. These parameters take effect for all gateways in the same CT/Engine address space.

The format of the KLGINGWY parameters is shown below. All parameters must be provided as a single logical statement.

```
[BCGROUP=bcgroup] [MONITOR=interval|5:00]
```

BCGROUP Identifies the broadcast group assigned to a user when a more specific broadcast group name cannot be acquired from an authorized application list (APPLIST) or a gateway default. If BCGROUP is omitted, a global broadcast group is not defined.

MONITOR Defines the interval that occurs between application status tests. When the interval expires, each application defined in an APPLDEF statement is interrogated by VTAM INQUIRE. The status returned is retained for display on the menu.

Specify the interval as a decimal number indicating the length of time in seconds, or as *hh:mm:ss.* If the interval is specified as zero, no status inquiry takes place. The default value is 5 minutes. If you set the interval to a smaller value, network performance may suffer.

The MONITOR initialization parameter is completely unrelated to the MONITOR operator command, which controls message types. For information on the MONITOR command, see the *Operator's Guide*.

Defining a Network Entry Point

The HOSTGATE command defines a network entry point, or *gateway* that controls entry into the VTAM network. Each gateway is associated with an application implicitly or explicitly selected by CL/GATEWAY users entering the network.

You can use HOSTGATE to define more than one gateway. *&rhilev*.RLSCMDS(KLGCHGGW) defines three default gateways: KLGICFG1, KLGICFG2, and KLGICFG3 (members in library *&rhilev*.RLSPARM). These gateways are initially opened by their VTAM ACB names KLSGW001, KLSGW002, and KLSGW003, as shown below.

```
HOSTGATE KLSGW001 CONFIG=KLGICFG1 BCGROUP=GROUP1 DIALOG=KLGATEWY ATTENTION=KLSPG08
HOSTGATE KLSGW002 CONFIG=KLGICFG2 BCGROUP=GROUP2 DIALOG=KLGATEWY ATTENTION=KLSPG08
HOSTGATE KLSGW003 CONFIG=KLGICFG3 BCGROUP=GROUP2 DIALOG=KLGATEWY
```

KLGICFG1 requires user ID and password validation for access to the Main Menu.

KLGICFG2 presents the menu directly to the user, without ID and password validation.

KLGICFG3 connects the user directly to the CT/Engine operator facility. No triggers or CT/Engine functions, such as ATTN, are available through KLGICFG3 unless the single session is a MULTI-type.

You can use HOSTGATE commands to define additional gateways, which can be set up in new configuration members in *&rhilev*.RLSPARM. For example, you might want one gateway to process all in-house terminals attached to dedicated lines, and another to control all terminals connected by a public data network. Each gateway can define unique panels, security validation techniques, and authorized application lists. You can use the APPLDEF command to define a gateway as an application.

To connect separate hosts or networks, define HOSTGATE commands in each host or network. The PASS dialog function can pass (VTAM CLSDST PASS) a physical or virtual terminal to a HOSTGATE on any host connected by cross-domain or cross-network facilities.

HOSTGATE commands are normally found in *&rhilev*.RLSCMDS(KLGCHGGW). This ensures that all gateways are available to users immediately after network startup. You can also issue HOSTGATE commands from the CT/Engine operator facility to add new gateways without recycling the CT/Engine address space.

If a gateway configuration needs to change while the product is running, implement the new configuration as follows:

1. Use the CT/Engine CLOSE command to shut down the gateway. For the CLOSE operand, specify the ACB name of the gateway to be stopped.
2. Issue a HOSTGATE command to restart the gateway with the new configuration.

For detailed information on the HOSTGATE command, see the *Operator's Guide*.

Gateway Configuration

Each gateway is defined by a number of predefined data items, called *data elements*. Data elements identify the user, handle the application selection process, and control a number of other session services.

The value associated with a data element may be derived from a variety of mechanisms, or *data sources*. A data source identifies a facility, such as a panel or a resolution routine, that acquires a data element value.

The collection of data elements and associated data sources is called *gateway* configuration. A gateway configuration consists of a series of entries, each containing the name of a data element followed by a list of one or more candidate data sources. Figure 12 shows the format of a data definition in a gateway configuration member.

```
data element        [USERDATA(position|keyword)]
                    [NAM]
                    [PANEL(panel)]
                      [DISPLAY]
                      [LIMIT(n)]
                      [PROMPT('string')]
                      [VARIABLE(varname)]
                    [EXIT(lmodname)]
                      [EXAMINE]
                    [DEFAULT(value)]
                    [OPTIONAL|REQUIRED|STATIC]
```

Figure 12. Data Definition Format

When several data sources are specified for a single data element, the candidates are examined in turn until a value is acquired. The sequence in which candidate data sources are examined is called the *acquisition sequence*. Dialog KLGDRES in library *&thilev*.TLSPNLS resolves the data elements.

The gateway configuration member is also the source for the messages presented during the logon and application selection dialogs. When an application selection error or other exceptional condition is encountered, the appropriate message appears on the user's screen.

Figure 13 on page 68 shows an example of a gateway configuration member, consisting of data definitions and messages.

```
USERID                                                  -
         REQUIRED                                       -
         USERDATA(0)                                    -
         PANEL(KLGLGON)                                 -
         PROMPT('ENTER USERID')
PASSWORD                                                -
         REQUIRED                                       -
         USERDATA(1)                                    -
         PANEL(KLGLGON)                                 -
         PROMPT('ENTER PASSWORD')                       -
         LIMIT(3)
NEWPSWD                                                 -
         REQUIRED                                       -
         USERDATA(NEWPSWD)                              -
         PANEL(KLGLGON)                                 -
         PROMPT('PASSWORD EXPIRED, ENTER NEW PASSWORD')
ACCOUNT                                                 -
         OPTIONAL                                       -
         PANEL(KLGLGON)
PROC                                                    -
         OPTIONAL                                       -
         PANEL(KLGLGON)
GROUP                                                   -
         OPTIONAL                                       -
         PANEL(KLGLGON)
APPLIST                                                 -
         OPTIONAL                                       -
         DEFAULT(&VSPAPLST)
DEST                                                    -
         REQUIRED                                       -
         USERDATA(SESSID)                               -
         PANEL(KLSVSEL)                                 -
         PROMPT(' ')                                    -
         MENU                                           -
         HELP(KLGWDEST)
LTERM                                                   -
         USERDATA(LTERM)                                -
         PANEL(KL1PGC9)                                 -
         PROMPT('ENTER LTERM')                          -
         HELP(KLGWLTRM)                                 -
         DEFAULT(&VIGUSER)
LOGMODE                                                 -
         OPTIONAL
POOL                                                    -
         DEFAULT(&DEFPOOL)
PRTLTERM                                                -
         PANEL(KLGDIMS)                                 -
         PROMPT('ENTER PRINTER LTERM')
PRTNODE                                                 -
         NAM
```

Figure 13. Gateway Configuration Member

Data Elements

Data elements provide the basis for the gateway's decisions. Each data element either identifies the user or conveys application selection and session establishment criteria.

For example, in Figure 13, the USERID, PASSWORD, and NEWPSWD data elements identify the user, current password, and new password, respectively. Once a value is obtained for each, the gateway can perform network access security validation. The recommendation returned by the security system determines whether network access is permitted.

Data Element Modifiers

For each data element, you can specify one of the following modifiers to determine how the gateway configuration processes the data element:

OPTIONAL The gateway does not require that the data element be resolved, even though it is specified as part of the gateway configuration.

REQUIRED Resolution (i.e., dialog KLGDRES) will not return without a value for the data element. An error condition will occur if none of the data sources provides a value for the data element.

STATIC The gateway acquires a value for a data element once only. This modifier is intended for the PANEL data source, to prevent a panel from being displayed more than once to acquire a value for the data element. When you specify STATIC for a PANEL data source, the gateway attempts to acquire the data element from the panel until the LIMIT threshold is reached. Once the data element is acquired, the gateway never attempts to acquire the data element again for the current gateway session.

If you do not enter a modifier, each data element specified in the configuration is assigned a default modifier. However, based on other parameters in the gateway configuration, the default may change from one gateway to another. The data elements are listed below in alphabetical order.

ACCOUNT Specifies the site account number assigned to the user. The Network Access Manager (NAM) passes the ACCOUNT data element to a security system or to the NAM user exit. Although the gateway does not enforce special formatting requirements, the format of the account number should be the same as required by MVS or your security system. The default modifier for ACCOUNT is OPTIONAL.

APPLIST Identifies an authorized application list maintained in *&thilev.*TLSPARM (for example, KLGIAPL1). If specified, this member identifies the subset of network applications available to the user.

DEST Identifies the selected application. It must correspond to the session ID identified in the DEST parameter of an APPLDEF command. The gateways that allow the user to select an application from a menu generally obtain DEST from the PANEL data source during an application selection dialog.

A destination name is always required. If DEST is omitted, it is assumed to be the first token of the userdata string.

The default modifier for DEST is REQUIRED. Since DEST must be acquired when it is defined as part of the configuration, a specification of OPTIONAL is invalid.

In order to properly resolve the DEST data element, the KLGICFG3 configuration member in your *&rhilev.*RLSPARM requires one of the following:

- PANEL(KLGDST)
- panel with similar logic, as described in the comments of dialog KLGDST

GROUP Identifies the group assigned to the user. The NAM facility passes the GROUP data element to a security system or to the NAM user exit. Although the gateway does not enforce special formatting requirements, the format of the group name should be the same as required by MVS or your security system. The default modifier for GROUP is OPTIONAL.

LOGMODE Specifies the VTAM logmode table entry name to be used in establishing a session between the user and the application. If specified, the LOGMODE data element overrides any logmode value designated in the gateway virtual terminal pool or the VTAM APPL statement. The default modifier for LOGMODE is OPTIONAL.

LTERM (CL/GATEWAY for IMS only) Identifies the IMS/DC logical terminal (LTERM) name to be associated with the user requesting access to an IMS/DC subsystem. The default modifier for LTERM is REQUIRED whenever the CL/GATEWAY for IMS user LTERM assignment session service is used. In this case, a specification of OPTIONAL will produce an error if a value for LTERM is not acquired.

NEWPSWD Supplies a 1- to 8-character string consisting of a user's new password. If provided, the value of NEWPSWD replaces the value of PASSWORD upon successful validation of the current USERID and PASSWORD. If NEWPSWD is not defined, the gateway cannot perform PASSWORD maintenance. The default modifier for NEWPSWD is OPTIONAL.

PASSWORD Supplies a 1- to 8-character string consisting of a personal identification code used to confirm the identity of the user specified by the USERID data element. The default modifier for PASSWORD is REQUIRED.

POOL Specifies a virtual terminal pool used to form a virtual session between the user and the application.

The name of the desired terminal pool is usually specified in the APPLDEF statement identifying the application. However, the POOL data element provides an alternate method for allowing dynamic selection of a virtual terminal pool on the basis of the user's terminal characteristics or other specified criteria. If you use the POOL data element to select a pool dynamically, you must include `POOL='*'` or `POOL='&DEFPOOL'` in the APPLDEF statement.

The default modifier for POOL is REQUIRED. Since this data element is resolved only if `POOL='*'` or `POOL='&DEFPOOL'` is included in the APPLDEF, OPTIONAL should never be specified.

PROC	Specifies a valid TSO logon procedure. The PROC data element is passed to the NAM user exit. Although the gateway does not enforce special formatting requirements, the format of the logon procedure name should be the same as required by MVS. The default modifier for PROC is OPTIONAL.
PRTLGMOD	Specifies the VTAM logmode table entry name to be used in establishing a session between the user and a physical printer. PRTLGMOD overrides the logmode specified in the VTAM definition for the printer logical unit. The default modifier is OPTIONAL.
PRTLTERM	(CL/GATEWAY for IMS only) Identifies an IMS/DC printer terminal LTERM associated with the user requesting access to an IMS/DC subsystem. The default modifier for PRTLTERM is REQUIRED whenever the CL/GATEWAY for IMS user LTERM assignment session service is requested and the PRTNODE data element is included in the configuration. In this case, a specification of OPTIONAL will produce an error if a value for PRTLTERM is not acquired.
PRTNODE	Identifies the printer terminal logical unit associated with the user or terminal accessing an application. It may be specified for any application that uses virtual printer terminal pooling. The default modifier for PRTNODE is REQUIRED whenever the CL/GATEWAY for IMS Session Cleanup or User LTERM Assignment session service is applied to both the user's terminal and an associated printer terminal. In this case, a specification of OPTIONAL will produce an error if a value for PRTNODE is not acquired. The PRINTER operand on the APPLDEF command must be set to OPTIONAL or REQUIRED for resolution to occur. If the operand is not specified, PRTLTERM resolution will not be driven.
PRTPOOL	Identifies a virtual printer pool used to form a virtual session between the user and the application. The name of the desired printer pool is usually specified in the APPLDEF statement identifying the application. However, the PRTPOOL data element provides an alternate method for allowing dynamic selection of a virtual printer pool on the basis of the user's session characteristics or other criteria. If you use the PRTPOOL data element to select a terminal dynamically, you must include **`PRTPOOL='*'`** or **`PRTPOOL='&DEFPOOL'`** in the APPLDEF statement. The default modifier for PRTPOOL is REQUIRED. Since this data element is resolved only if **`PRTPOOL='*'`** or **`PRTPOOL='&DEFPOOL'`** is included in the APPLDEF, OPTIONAL should never be specified.
USERDATA	Contains an application-dependent character string that is passed to the application when a session is established with the user. Defined USERDATA data elements consisting of nulls or blanks are not sent to the application. USERDATA can also identify the name of the destination application when the DEST data element is not defined. In this case, the first token in the userdata string is assumed to contain the

application name. The remainder of the string is sent to the application as userdata.

You can also specify USERDATA in an APPLDEF command. If you do so, the USERDATA specified in APPLDEF overrides the USERDATA data element.

The default modifier for USERDATA is OPTIONAL. However, if the DEST data element is not defined in the configuration, the USERDATA data element must be defined. In that case, the default modifier is REQUIRED.

USERID
Supplies a 1- to 8-character string that uniquely identifies the user. USERID and PASSWORD are used in combination to provide the basis for the network access authorization validation supported by CL/GATEWAY and by external security systems, including RACF™, CA-ACF2®, and CA-TOP SECRET®.

Typically, the USERID, PASSWORD, and NEWPSWD data elements are obtained from a gateway logon or application selection dialog. That is, the PANEL data source is usually specified for those data elements. However, when the gateway is used in a transparent mode without an application selection dialog, these elements can be passed to the gateway as userdata to provide network access security or eliminate redundant logon sequences in the accessed application.

When NAM is specified as a data source for any data element (such as APPLIST or LTERM), the USERID data element identifies the major key of the desired data element variable. That is, USERID is the key for NAM lookups for any data element NAM resolves.

The default modifier for USERID is REQUIRED.

Data Sources

A *data source* is the place where the gateway searches for a data element. Each data source identifies a facility, such as the NAM database or a dialog panel, that is responsible for acquiring a data element value.

Several data sources may be designated for a single data element. When several data sources are specified, they are examined in a predefined order to acquire the data element value. This sequence is called the *resolution sequence*. When a data source is not specified for a data element, the data source is not examined. The data sources are listed here in resolution sequence order.

USERDATA(position|keyword)
Specifies that the data element may be included in the userdata string passed to the gateway by VTAM. A USERDATA parameter identifies the manner in which the data element is extracted from the userdata string.

NAM
Specifies that the data element is to be extracted from the NAM database record identified by the USERID data element. The NAM data source can be used only when a NAM database has

been allocated and loaded with user records. In addition, NAM may be specified only for data elements maintained in NAM records.

PANEL(panel)	Specifies a dialog panel definition that may be used to obtain a value for the data element. A named member of the panel library is invoked and is responsible for setting the appropriate variable. Note that a panel does not have to be interactive; i.e., it can run the dialog and quit. If the DISPLAY data source qualifier is included, the dialog routine can inspect and replace a data element value previously obtained from another data source. Several data elements may be acquired concurrently from the same panel. ***Caution:*** The member name specified should omit the National Language suffix; e.g., use KLGLGON, not KLGLGON1.
EXIT(lmodname)	Identifies an exit routine that may be invoked to derive a value for the data element. If the EXAMINE data source qualifier is included, the exit routine can inspect and replace a data element value previously obtained from another data source.
DEFAULT(value)	Identifies a default value that is assigned to the data element when no other specified data source can produce an appropriate value. When used alone, the DEFAULT data source performs an unconditional assignment of a value to a data element.

USERDATA Data Source

The USERDATA data source indicates that a userdata string passed to the gateway may contain a value for a given data element. The USERDATA operand indicates either the relative position of the data element value, or identifies a keyword from which the appropriate value can be extracted.

In the *relative position* format, the USERDATA operand is specified as a decimal number. The token found in the userdata string at the zero-relative position is extracted and assigned to the data element. If the userdata string is null or contains fewer tokens than required, no value assignment is made. This format is frequently used when CL/GATEWAY is operating in transparent mode, particularly when another logon solicitor is in use or when control is passed from one gateway to another. Data obtained from the logon solicitor or other VTAM application may be conveniently passed to the gateway through a series of userdata tokens arranged in a predefined sequence.

In the *keyword* format, the USERDATA operand is specified as a character string that corresponds to a keyword occurring in the userdata string. When located, the value associated with the keyword is assigned to the data element. If the userdata string is null or does not contain the specified keyword, no value assignment is made. This format can provide a quick logon capability, bypassing the logon and application selection dialogs. If the user is able to satisfy all data element requirements by including the specified keywords in the logon userdata string, no additional prompting need occur.

In the first two data definition statements shown in the example below, the USERID and PASSWORD data elements may be supplied to the gateway as the first and second tokens occurring in the userdata string. If a userdata string is not furnished, both data elements must be acquired from other data sources.

The DEST statement illustrates the manner through which a keyword in the userdata string can be defined. If present, the value associated with the SENDTO keyword is assigned to the DEST data element. If a userdata string is not furnished, DEST must be acquired from another data source.

```
USERID      USERDATA(0)           USERID COMES FIRST
PASSWORD    USERDATA(1)           FOLLOWED BY PASSWORD
*
DEST        USERDATA('SENDTO')    EXTRACT SENDTO=dest
```

NAM Data Source

You can use NAM and its database to enroll authorized network users. The NAM database can perform basic security functions that are not provided by an external security system.

NAM records also provide a convenient means of associating a variety of data elements directly to an individual user or physical terminal. All data elements except USERID may be resolved from the NAM database. For the NAM database to resolve data elements, the USERID data element must be specified in the gateway configuration member.

Data element values are stored and retrieved in the NAM database using the data element name as the variable name.

In the example below, authorized application lists have been defined for a subset of its users. The USERID is the first data element the gateway obtains. Because the APPLIST data element specifies the NAM data source, NAM uses the USERID as the major key and the data element name as the minor key to retrieve the appropriate user record from the database. The APPLIST member name contained in the record is extracted and assigned to the APPLIST data element. Note that the NAM keyword has no operands.

```
APPLIST  NAM   /* AUTHORIZED APPLICATION LIST */
```

PANEL Data Source

The PANEL data source provides a mechanism for acquiring data elements directly from the user. The operand of the PANEL keyword identifies the name of a panel definition residing in the panel library (*&rhilev.*RLSPNLS). If the data element has not already been resolved and the data element is not optional, the panel is invoked. If the data element is optional, you can force display of the panel by specifying DISPLAY in the data definition.

When a dialog is invoked, it sets the variables associated with the data elements that the panel will display and/or acquire. Table 5 on page 75 shows the variable name associated with each data element.

Table 5. Data Element Variable Names

Data Element	Variable Name	Variable Scope
ACCOUNT	VIGACCT	session
APPLIST	VIGAPLST	shared
DEST	VIGDEST	shared
GROUP	VIGGROUP	session
LOGMODE	VIGLMODE	shared
LTERM	VIGLTERM	shared
NEWPSWD	VIGNPSWD	session
PASSWORD	VIGPSWD	session
POOL	VIGPOOL	shared
PROC	VIGPROC	session
PRTLGMOD	VIGPRTLG	shared
PRTLTERM	VIGPLTRM	shared
PRTNODE	VIGPNODE	shared
PRTPOOL	VIGPRTPL	shared
USERDATA	VIGDATA	shared
USERID	VIGUSER	session

Each data element that includes the PANEL data source can specify a common panel name. The user can enter several data elements on the same panel display. Alternatively, a distinct panel can be provided for each data element. In this case, the user is presented with a series of panel displays, one for each data element that requires resolution.

You can modify the operation of the PANEL data source by including one or more of the data source modifiers described below.

DISPLAY Specifies that the panel used to collect the data element will be displayed regardless of whether a value has been acquired from another data source. If a value is associated with the data element, it is also presented. When DISPLAY is omitted, the panel is presented only if a value has not already been acquired for the data element. The DISPLAY keyword has no operands.

LIMIT(n) When a panel is presented to acquire a data element from the user, the panel is usually redisplayed when the element is not entered correctly. However, when the user performs a logon sequence involving the USERID and PASSWORD data elements, it is desirable to establish a ceiling on the number of logon attempts. The LIMIT qualifier may be used to specify the maximum number of attempts allowed. If the threshold is exceeded, the session with the gateway user is dropped. Specify the LIMIT operand as 0, 1, 2, or 3. For example:

```
LIMIT(2)
```

Every attempted resolution, whether valid or invalid, of a data element counts against the LIMIT for that data element. Null data also counts against the LIMIT. If a data element with a low LIMIT is resolved because of failures to resolve a data element with a higher LIMIT, the logon is rejected.

PROMPT('string') Associates a one-line prompt or other directive with the panel display. When a panel is presented to resolve a specific data element, the prompt is also displayed. The location of the prompt is determined by the presence of the VIGMSG session variable.

Specify the prompt as a character string enclosed in quotes. For example:

```
PROMPT('Enter application name from menu')
```

Some data elements have default prompts that will appear on the screen if a prompt is not specified. For example, the default prompt for USERID is

```
'PLEASE ENTER USER ID'
```

If you do not want a prompt to appear on your users' screens, enter blanks for the character string:

```
PROMPT('      ')
```

VARIABLE(varname) Reassigns the variable name that represents a specific data element in a panel definition. Each data element that can be acquired by the PANEL data source is associated with a default variable name. If these defaults are not suitable, you can use the VARIABLE qualifier to define a new variable.

Specify the VARIABLE operand as a variable name not longer than 8 characters. Note that the variable prefix character (&) is omitted. For example:

```
VARIABLE(MYDEST)
```

EXIT Data Source

The EXIT data source extends the data element resolution process to include data sources and data acquisition techniques that you define. The EXIT keyword includes an operand that specifies the name of the data element resolution routine you supply.

You can modify the operation of the EXIT data source by including the EXAMINE qualifier.

EXAMINE Specifies that the resolution routine is to be given control to examine or modify the corresponding data element, regardless of whether a value has been acquired from another data source. When EXAMINE is omitted, the resolution routine is invoked only if a value has not already been acquired. The EXAMINE keyword has no operands.

An example of a site-written resolution routine follows. The example is distributed in *&thilev.*TLSSAMP(KLGXELEM).

```
V147     TITLE 'V147 SAMPLE DATA ELEMENT RESOLUTION EXIT'
         KLV$$MAC ,
         MACRO
&L       #TXT  &TEXT
         LCLA  &LEN
&LEN     SETA  K'&TEXT-2
&L       DC    AL1(&LEN)
         DC    C&TEXT
         MEND
*
KLGXELEM CSECT
*
*----------------------------------------------------------------
*        REGISTER EQUATES.
*----------------------------------------------------------------
         YREGS ,
         ACFREGS LINK=R14
RLSA     EQU   R13
RSEC     EQU   R12
REPL     EQU   R11
RAC2     EQU   R10
RDEL     EQU   R9
RLEN     EQU   R8
RBAL     EQU   R7
*
*----------------------------------------------------------------
*        RETURN CODE VALUES.
*----------------------------------------------------------------
RC00     EQU   00                   REQUEST IGNORED, CONTINUE
RC04     EQU   04                   ACF2 HAS ALLOWED SYSTEM ACCESS
RC08     EQU   08                   ACF2 REJECTED THE SUPPLIED USERID
*
         SPACE 1
         USING SAVE,RLSA
         USING KLGXELEM,RSEC
         USING #EP,REPL
         SPACE 1
*
*-----------------------------------------------------------------------
*        DECLARE WORK AREA SIZE
*-----------------------------------------------------------------------
         NOP   L'SAVE               DECLARES WORK AREA SIZE
*
         STM   R14,R12,12(R13)      SAVE CALLERS REGISTERS
*
```

```
*----------------------------------------------------------------------
*        ESTABLISH MODULE AND #EPL BASE
*----------------------------------------------------------------------
         LR    RSEC,R15            ESTABLISH MODULE BASE
         LR    REPL,R1             ESTABLISH #EPL BASE
*
*----------------------------------------------------------------------
*        CREATE LINKAGE AND ESTABLISH SAVE AREA BASE
*----------------------------------------------------------------------
         L     R1,#EPAREA          SUPPLIED WORKAREA ADDRESS
         ST    R13,4(,R1)          REVERSE LINKAGE
         ST    R1,8(,R13)          FORWARD LINKAGE
         LR    RLSA,R1             ESTABLISH WORKAREA BASE
*
*----------------------------------------------------------------------
*        PROCESS REQUEST
*----------------------------------------------------------------------
         LA    R15,RC00            DEFAULT TO SUCCESSFUL RC
         SLR   R1,R1               DEFAULT TO NO MESSAGE
         CH    R0,=Y(MAXREQ)       REQUEST SUPPORTED ?
         BH    EXIT                NO - JUST EXIT
         LR    R2,R0               GET REQUEST IN USABLE REG
         B     REQUEST(R2)         PROCESS REQUEST
*
*----------------------------------------------------------------------
*        REQUEST BRANCH TABLE
*----------------------------------------------------------------------
REQUEST  DS    0H
         B     USERID              USERID REQUEST
         B     PASSWORD            PASSWORD REQUEST
         B     NEWPSWD             NEW PASSWORD REQUEST
         B     APPLIST             APPLIST REQUEST
         B     DEST                DESTINATION REQUEST
         B     PRTNODE             PRTNODE REQUESTS
         B     PRTLTERM            PRTLTERM REQUESTS
         B     LTERM               LTERM REQUESTS
         B     USERDATA            USERDATA REQUEST
         B     LOGMODE             LOGMODE REQUEST
         B     POOL                POOL REQUEST
         B     PRTPOOL             PRINTER POOL REQUEST
         B     GROUP               GROUP REQUEST
         B     ACCOUNT             ACCOUNT REQUEST
         B     PROC                PROC REQUEST
MAXREQ   EQU   *-REQUEST
*
USERID   DS    0H
         B     EXIT                RETURN TO CALLER
PASSWORD DS    0H
         B     EXIT                RETURN TO CALLER
NEWPSWD  DS    0H
         B     EXIT                RETURN TO CALLER
APPLIST  DS    0H
         B     EXIT                RETURN TO CALLER
DEST     DS    0H
```

```
         B     EXIT                RETURN TO CALLER
PRTNODE  DS    0H
         B     EXIT                RETURN TO CALLER
PRTLTERM DS    0H
         B     EXIT                RETURN TO CALLER
LTERM    DS    0H
         BAL   RBAL,ACF2INIT
         L     RDEL,#EPSTR         POINT AT DATA ELEMENT
* /* USER */
******   CHANGE THE FOLLOWING IC TO EXTRACT THE LTERM NAME LENGTH
******   FROM THE LID RECORD.
*
         SR    RLEN,RLEN           CLEAR FOR IC
         IC    RLEN,LID????L       GET LTERM LENGTH FROM LIDREC
         C     RLEN,#EPSTRLN       IS IT TOO LONG ?
         BH    ERRLTERM            YES ... ISSUE A MESSAGE
         STC   RLEN,0(,RDEL)       SET LENGTH
         BCTR  RLEN,0              REDUCE FOR ...
         EX    RLEN,MVCLTERM       COPY LTERM NAME FROM LIDREC
*
         LA    R15,RC00            SET SUCCESSFUL RC
         SLR   R1,R1               SHOW NO MESSAGE
         B     EXIT                RETURN TO CALLER
ERRLTERM $
         LA    R15,RC04            SET TERMINATE RC
         LA    R1,MSGNOLTM         ADDRESS RETURN MESSAGE
         B     EXIT                RETURN TO CALLER
* /* USER */
******   CHANGE THE FOLLOWING MVC TO COPY THE LTERM NAME FROM
******   THE LID RECORD.
*
MVCLTERM MVC   1(*-*,RDEL),LID????? *** EXECUTED ***
USERDATA DS    0H
         B     EXIT                RETURN TO CALLER
LOGMODE  DS    0H
         B     EXIT                RETURN TO CALLER
POOL     DS    0H
         B     EXIT                RETURN TO CALLER
PRTPOOL  DS    0H
         B     EXIT                RETURN TO CALLER
GROUP    DS    0H
         B     EXIT                RETURN TO CALLER
ACCOUNT  DS    0H
         B     EXIT                RETURN TO CALLER
PROC     DS    0H
*        B     EXIT                RETURN TO CALLER
EXIT     DS    0H
         L     R13,4(,R13)         RESTORE PRIOR R13
         L     R14,12(,R13)        RESTORE R14
         L     R0,20(,R13)         RESTORE R0
         LM    R2,R12,28(R13)      RESTORE R2 - R12
         BR    R14                 RETURN TO CALLER
         TITLE 'ACF2 INITIALIZATION'
ACF2INIT DS    0H
*
```

```
*---------------------------------------------------------------
*        FIND ACF2 CVT
*---------------------------------------------------------------
      ACFINCVT RAC2,              LOCATE ACF2 CVT                -
               NONE=ACF2DOWN
*
*---------------------------------------------------------------
*        INITIALIZE ACF2 PARAMETER BLOCK
*---------------------------------------------------------------
         XC    ACVALD(ACVLEN),ACVALD   CLEAR ACF2 SVCA PARM BLOCK
         MVI   ACVFCN,1           RETRIEVE LIDREC FUNCTION
         MVI   ACVSFCN,ACVSINFO   RETRIEVE LIDREC FUNCTION
         LA    R0,VMSGAREA        MESSAGE BUFFER RETURN AREA
         ST    R0,ACVMSG          ... SAVE IN PARM BLOCK
         LA    R0,LIDREC          LOGONID RECORD RETURN AREA
         ST    R0,ACVRECB         ... SAVE IN PARM BLOCK
         LA    R0,LIDLEN          LOGONID RECORD RETURN AREA LENGTH
         ST    R0,ACVRECL         ... SAVE IN PARM BLOCK
         LA    R0,UIDAREA         UID STRING RETURN AREA
         ST    R0,ACVUIDB         ... SAVE IN PARM BLOCK
         LA    R0,L'UIDAREA       UID STRING RETURN AREA LENGTH
         ST    R0,ACVUIDL         ... SAVE IN PARM BLOCK
*
*---------------------------------------------------------------
*        GET USERID FROM #EPL
*---------------------------------------------------------------
         LA    R15,RC04           BAD USERID RC
         LA    R1,MSGNOUSR        NO USERID MESSAGE
         ICM   R2,15,#EPUSER      LOCATE USERID
         BZ    EXIT               RETURN IF NO USERID
         SLR   R3,R3              ZERO FOR INSERT
         ICM   R3,1,0(R2)         GET LENGTH
         BZ    EXIT               RETURN TO CALLER IF NOT SUPPLIED
         MVC   ACVLID,=CL8' '     CLEAR FIELD
         BCTR  R3,0               EXECUTE LENGTH
         EX    R3,MVCID           MOVE PARAMETER
*
*---------------------------------------------------------------
*        RETRIEVE LID RECORD FROM ACF2
*---------------------------------------------------------------
        ACFSVC ACVALD,            RETRIEVE LIDREC                -
               NONE=ACF2DOWN,                                    -
               CVT=HAVE
         LTR   R15,R15            SUCCESSFUL ?
         BZR   RBAL               YES - RETURN
         LA    R15,RC04           TERMINATE RETURN CODE
*
*---------------------------------------------------------------
*        RETRIEVE ACF2 ERROR MESSAGE (IF ANY), SET RETURN CODE,
*        AND RETURN TO CALLER.
*---------------------------------------------------------------
ACVEXIT  DS    0H
         L     R2,ACVMSG          MESSAGE AREA ADDRESS
         SLR   R1,R1              ASSUME NO MESSAGE
         SLR   R3,R3              ZERO FOR INSERT
         ICM   R3,3,0(R2)          ANY MESSAGE RETURNED BY ACF2 ?
         BZ    EXIT                NO - RETURN TO CALLER
         SH    R3,=H'4'            ADJUST FOR RDW
         BNP   EXIT                NO - RETURN TO CALLER
         STC   R3,3(,R2)           SHOW NEW LENGTH
         LA    R1,3(,R2)           ADDRESS MESSAGE
         B     EXIT                NO - RETURN TO CALLER
MVCID    MVC   ACVLID(*-*),1(R2)   *** EXECUTED ***
*
*---------------------------------------------------------------
*        IF ACF2 IS DOWN, DENY ACCESS TO ALL USERS.
*---------------------------------------------------------------
ACF2DOWN DS    0H
         LA    R15,RC04            TERMINATE RETURN CODE
         LA    R1,MSGNOACF         RETURN MESSAGE
         B     EXIT                DONE
         TITLE 'MESSAGES'
MSGNOACF #TXT  'ACF2 NOT INSTALLED OR NOT AVAILABLE, ACCESS DENIED'
MSGNOUSR #TXT  'USERID NOT SUPPLIED, ACCESS DENIED'
MSGNOLTM #TXT  'NO LTERM FOUND FOR THIS USER, ACCESS DENIED'
*
         TITLE 'LITERAL POOL'
         LTORG ,
         TITLE 'STORAGE AREA'
**********************************************************************
*
*        THIS IS THE STORAGE AREA THAT WILL BE ALLOCATED BY THE
*        NAM INTERFACE FOR USE BY THIS ROUTINE TO ENSURE THAT
*        THE ROUTINE IS REENTRANT. IT CONSISTS OF A STANDARD OS
*        SAVE AREA FOLLOWED BY ANY LOCAL STORAGE THE EXIT REQUIRES.
*
**********************************************************************
#SAVE    DSECT
         DS    19F                 STANDARD SAVE AREA
UIDAREA  DS    CL24                ACF2 UID STRING RETURN AREA
VMSGAREA DS    CL128               ACF2 ENTRY VALIDATION MESSAGES
*
         EJECT ,
         ACVALD DSECT=NO           SYSTEM ENTRY VALIDATION PLIST
         EJECT ,
         LIDREC DSECT=NO           ACF2 LOGONID RECORD FORMAT
SAVE     EQU   #SAVE,*-#SAVE       INCLUSIVE LABEL
*
         EJECT ,
         ACCVT ,
         END   ,
```

The sample exit routine illustrates coding guidelines and highlights basic concepts. KLGXELEM calls TGMACLIB(KLG#GEPL), which provides a parameter list for the exit. Most parameters in the list point to data elements to be resolved during the exit.

Parameter	*Points to:*
AREA	User work area address
STR	String address
STRLN	String length
PHYSN	Physical node name
VIRTN	Virtual node name
APPL	Application name
USER	USERID string pointer
APLST	APPLIST string pointer
PNODE	PRTNODE string pointer
PLTRM	PRTLTERM string pointer
LTERM	LTERM string pointer
DEST	DEST string pointer
LMODE	LOGMODE string pointer
DATA	USERDATA string pointer
POOL	POOL string pointer
IBUSD	Inbound USERDATA string pointer
GROUP	GROUP string pointer
ACCT	ACCOUNT string pointer
PROC	PROC string pointer
DIALG	Dialog service
AUB	Active user block

Some exit parameters point to a length followed by the data; these include APLST, DATA, DEST, IBUSD, LMODE, LTERM, PLTRM, PNODE, POOL, and USER. Other exit parameters point to an 8-character string (padded with blanks on the right, if necessary); these include APPL, PHYSN, and VIRTN.

In coding your own exit routines, keep in mind the following sequence of data element resolution:

- USERID is always the first element resolved.
- APPLIST must be resolved before DEST.
- DEST must be resolved before LOGMODE, LTERM, POOL, PRTLTERM, PRTNODE, and USERDATA.

You can invoke KLV$$MAC of TGMACLIB at the beginning of any exit and then use the $USREXIT macro to get variables, manage memory, and use dialog services. An MVS LOAD can also be performed during an exit, provided there is enough RESERVE space available in the address space. See “RESERVE” on page 199.

In the example shown below, the site has supplied a resolution routine for the APPLIST data element. The USERDATA data source indicates that an APPLIST designation may also be available in the userdata string passed to the gateway. The EXAMINE qualifier allows the resolution routine to inspect and/or replace the APPLIST name, even if the APPLIST name was provided in the userdata string.

```
APPLIST    USERDATA('APPNAME')   EXIT(APPEXIT)   EXAMINE
```

For an example of a destination exit, see *&rhilev*.RLSSAMP(KLSXMSGS).

DEFAULT Data Source

The DEFAULT data source provides a value for a data element when no other data source has been defined, or when defined data sources have failed to resolve the data element. The DEFAULT data source is examined last in the resolution sequence. Therefore, all other data sources that can supply a value take precedence over DEFAULT.

The value specified in the DEFAULT keyword operand is assigned to the data element before resolution is attempted. Therefore, if EXIT is specified, the resolution routine observes the default value in the storage cell. Similarly, if a PANEL data source is specified, the panel displayed presents the default value for inspection or modification.

Suppose we use the following example:

```
DEST        DEFAULT(SYSIMS)
APPLIST     NAM DEFAULT(DFTLIST)
LTERM       DEFAULT(&SYSTERM)
```

The first line assigns a default value to the destination application (DEST). Since no other data source is defined, the DEST data element is always acquired from the specified default value. In this example, the SYSIMS destination application name is assigned to the DEST data element for each user coming through the gateway.

The second line selects the DFTLIST authorized application list for all users of the gateway, but only if a NAM record for the APPLIST cannot be obtained.

The third line gives to the LTERM data element the value of the variable &SYSTERM. The &SYSTERM variable is the logical unit name of the terminal. This example demonstrates a simple way of assigning an LTERM based on the physical terminal name.

Resolution Sequence

The dialogs that run when a terminal signs on (beginning with dialog KLGATEWY) *resolve,* (assign values to) the various data elements. The sequence begins by resolving the data element that identifies the user. It continues by examining data sources to resolve information about the various selections made.

The KLGDRES dialog implements the resolution sequence by invoking the data sources defined in the gateway configuration members KLGICFG1, KLGICFG2, and KLGICFG3 of *&rhilev*.RLSPARM.

Note: Candle does not recommend changing the KLGDRES dialog.

The following list summarizes the resolution sequence:

1. USERDATA is inspected, if it was supplied at logon.
2. If information from USERDATA did not resolve the data element, NAM is searched.

3. Panel sources (if any) are searched if either of the following applies:
   - Neither NAM nor USERDATA supplied a value.
   - The DISPLAY keyword is specified in the configuration member to force panel invocation.
4. The EXIT data source is inspected. An EXIT source is not invoked if a value is supplied by any of the prior sources, unless the EXAMINE keyword is included in the configuration member to force EXIT invocation.
5. The data element is set to the default when no other data source can produce a value.

Table 6 lists the standard dialogs, data elements, and variable names involved in the resolution process.

Table 6. Data Element Resolution

Standard Dialog	Data Element	Variable Name
KLGNTRY	USERID	VIGUSER
KLGNTRY	ACCOUNT	VIGACCT
KLGNTRY	GROUP	VIGGROUP
KLGNTRY	PROC	VIGPROC
KLGNTRY	NEWPSWD	VIGNPSWD
KLGNTRY	PASSWORD	VIGPSWD
KLSUINI1	APPLIST	VIGAPLST
KLSCNTL	DEST	VIGDEST
KLGSSHG	USERDATA	VIGDATA
KLGSSHG	LOGMODE	VIGLMODE
KLGSSHG	POOL	VIGPOOL
KLGSSHG	PRTNODE	VIGPNODE
KLGSSHG	BRTPOOL	VIGPRTPL
KLGSSHG	LTERM	VIGLTERM
KLGSSHG	PRTLTERM	VIGPLTRM
KLGSSHG	PRTLGMOD	VIGPRTLG

When a user logs on (see Figure 27 on page 235), the KLGATEWY dialog (supplied by Candle) starts.

KLGGW2 calls the KLGNTRY dialog, which processes the data elements concerned with user identification and logon procedure information (see Table 6). The KLSPG00 dialog then issues either the VIGENTRY CL/GATEWAY function, and/or the VSSENTRY CL/SUPERSESSION function (for those who have installed CL/SUPERSESSION). One of those functions invokes the KLSUINI1 dialog as the initialization dialog and KLSCNTL as the control dialog. KLSUINI1 sets up the names of the commands, selection characters, and status codes for the user's environment. It also sets up a list of all the applications that a user is authorized to access. KLSUINI1 resolves the APPLIST data element (see Table 6).

KLSCNTL (called from either VIGENTRY or VSSENTRY in the KLSPG00 dialog after KLSMP230 completes) clears out the DEST data elements and calls the KLGDRES dialog to resolve them. In the KLGICFG1 configuration, KLGDRES calls the KLSVSEL dialog, which displays the selection menu. KLSVSEL1 waits for a user action. When a selection is made, KLSCNTL calls dialog KLGSSHG, which completes the data element resolution.

Customizing Messages

You can customize the messages users see during logon and application selection by modifying the gateway configuration member.

When exceptions are detected, the gateway generates message numbers. Once identified, the message number is used to locate the message text defined by the MESSAGE statement in the configuration member. If no MESSAGE statement is found to match the message number, a default message is provided. The VIGMSG variable is then replaced by the message text, and the panel is redisplayed.

Here is the format of the MESSAGE statement:

```
MESSAGE  msgnumber  [msg_identifier|'msg text']
```

msgnumber The message number internally associated with an exception detected by a gateway. A description of the exceptions defined and the associated message numbers appears below.

Msg number	*Error or exception*
00	An application has been selected, but the gateway configuration has changed. The user is given a chance to respecify his request.
01	The requested application does not match any authorized session ID defined in an application definition (APPLDEF), or is not included in the authorized application list (APPLIST) associated with the user.
02	HELP was requested, but no HELP is available. The HELP operand can be specified in the APPLDEF command that defines the application, or in the configuration definition for a specific data element.
03	A valid destination was selected, but the destination application is inactive or is not accepting logon requests.

04	An undefined logmode name was specified. For virtual sessions, make sure the LOGMODE parameter on the VSM command specifies a valid entry name. If the &VIGLMODE product variable is being used to supply the logmode name, make sure it contains the correct name.
05	The destination application rejected the session request because the logical unit was not known. For example, an attempt to start a session with IMS will generate this message if the LU name of the physical terminal for CLSDST PASS sessions or the virtual terminal for SINGLE sessions does not define a valid IMS PTERM.
06	The logical unit rejected the session request because of unsupported session parameters. For example, an attempt to bind a 3278 model 2 with a model 3 bind will generate this message.
07	Unable to establish a virtual terminal session.
08	The virtual terminal pool specified for a SINGLE session is empty.
09	The virtual printer pool specified for a SINGLE session is empty.
10	The name of the application list is not found. No APPLIST command matches the name specified in the APPLIST data element.
11	Gateway setup error. A gateway configuration contains inconsistent definitions. This occurs, for example, if the IMS ASSIGN session service is selected but the LTERM data element is not defined as part of the configuration.
12	An unsupported command or function key was entered.
13	The printer terminal identified by the PRTNODE data element could not be acquired.
14	A virtual printer terminal session could not be established.

15	(CL/GATEWAY for IMS only) No IMS operator session is active with the required IMS subsystem. When either the ASSIGN or the DEQUEUE session service is selected, a virtual IMS master terminal (MTO) operator session must be available.
16	(CL/GATEWAY for IMS only) An undefined LTERM name has been specified. The name associated with the LTERM data element is not defined in the IMS/DC gen.
17	(CL/GATEWAY for IMS only) An undefined PTERM name has been specified. The logical unit name of the VTAM terminal is not defined in the IMS/DC gen. If terminal pooling is being used, this name will be the LU name of the virtual terminal.
18	(CL/GATEWAY for IMS only) The requested LTERM is not available. The LTERM is a valid LTERM name, but the ASSIGN command could not assign this LTERM to the session.
19	(CL/GATEWAY for IMS only) The STOP NODE command failed. This command is issued by a virtual MTO session.
20	(CL/GATEWAY for IMS only) The DEQUEUE NODE command failed. This command is issued by a virtual MTO session.
21	(CL/GATEWAY for IMS only) The RESTART NODE command failed. This command is issued by a virtual MTO session.
22	(CL/GATEWAY for IMS only) The START NODE command failed. This command is issued by a virtual MTO session.
23	(CL/GATEWAY for IMS only) The requested LTERM is in use. This message can occur if the specified LTERM is already in an active session, and the Extended LTERM Verification (XLV) option is requested.
24	No authorized applications.
25	Resolution limit exceeded.

msg_identifier	A character string beginning with the characters SMSGGW and containing no imbedded blanks. The message identifier is defined in a dialog named KLSMSGGn (where *n* is a 1-digit national language identifier).
msg text	A valid string expression up to 255 characters in length, to notify the user of the problem or exception.

By default, the tilde character (˜) translates to a single quote mark (') when a message is displayed. If you want to send and receive the tilde character in your messages, modify dialogs KLGMSG1E and KLGMSG2P by replacing the tilde with a character that you do not intend to use in messages.

Using &SYSPANEL to Process Gateway Errors

You can specify the &SYSPANEL function in the message text of any gateway message number, to provide the name of a panel that will go into effect whenever the exception associated with the message occurs. For example

```
MESSAGE 1 '&SYSPANEL(ERROR01)'
```

The &SYSPANEL method of processing configuration errors can be extremely valuable in

- configurations that do not use the PANEL data source for acquiring values for data elements such as DEST
- configurations with dialogs that do not contain a body section

In such configurations, the VIGMSG session variable cannot be used to display configuration error messages, since no screen image is available to display the message. The panel named in the &SYSPANEL function can be used to display error messages and to prompt for additional information. The prologue and epilogue sections of the dialog can perform additional processing, such as setting values for session variables or logging off the gateway.

Defining Applications

The APPLDEF command defines the applications that are accessible from all gateways. When you define applications with APPLDEF, they are all available to all users. If you are using an application selection panel, CL/GATEWAY presents all applications on that panel.

Optionally, you can associate an authorized application list (APPLIST) with the user or gateway to restrict application access.

Candle recommends that you include in a CLIST—for example, *&rhilev*.RLSCMDS (KLSCAPLS)—all APPLDEF commands that define production applications. However, you can also define applications in profiles. See the *Basic Configuration Guide* for instructions.

The session ID specified in the APPLDEF command creates a unique application definition. Therefore, any reference to the application in other commands must specify the session ID.

APPLDEF

Purpose

Defines an application that is accessible through CL/SUPERSESSION and CL/GATEWAY.

Context

CL/GATEWAY operator command

Format

APPLDEF sessid
DEST=destid
[ALTDEST=altdid]
[COMPRESS=YES|<u>NO</u>|IGNORE]
[DESC='string-exp']
[GROUP=nnn|<u>0</u>]
[HELP=member]
[IMS=DEQUEUE|ASSIGN|'ASSIGN,DEQUEUE'|<u>NO</u>]
[INITDLG=dlgname]
[LOGON='string-exp']
[MESSAGE='string-exp']
[MULTSESS=<u>YES</u>|NO]
[NEWGROUP=nnn]
[NOLIST]
[ORDER=nnn|<u>0</u>]
[POOL=vsmpool|'*']
[PRINTER=REQUIRED|OPTIONAL|<u>NONE</u>]
[PRTPOOL=vsmpool|'*']
[REMOVE]
[SIMLOGON=YES|<u>NO</u>|'data']
[TERMDLG=dlgname]
[USERDATA='string-exp']

Parameters

sessid	Specifies the name that displays on the Main Menu for the application identified by the DEST parameter. The session ID is limited to 8 characters.

The session ID creates a unique application definition. Therefore, any reference to the application in other commands must specify the session ID.

DEST	Identifies the applid served by the APPLDEF. If that application is not available, the gateway establishes a session with the alternate destination designated by ALTDEST. The DEST parameter is required. CL/GATEWAY for IMS only. When CL/GATEWAY for IMS session services are specified by the IMS=DEQUEUE/ASSIGN parameter, the DEST parameter identifies the name of an IMS environment definition statement.
ALTDEST	Identifies the secondary alternate session ID served by the APPLDEF. If the DEST application is not available, the gateway establishes a session with the ALTDEST application. The ALTDEST application typically identifies another copy of the application running on a backup system. The ALTDEST parameter is optional.
COMPRESS	Determines whether to use compression for this application. **YES** The session is eligible for both inbound and outbound compression, depending on the user, group, or global profile setting. **NO** The session is not eligible for compression. This is the default. **IGNORE** Leaves in effect the default you establish for the user by executing the administrator functions. COMPRESS is a valid parameter only if MULTSESS=YES is specified.
DESC	Describes the application or clarifies its use for display on the Main Menu. The maximum length of the description is 32 characters.
GROUP	Identifies a related group of applications. CL/GATEWAY displays all applications with an identical group number together on the Main Menu. By default, each application group occupies a different menu panel, and the groups are displayed in descending order. If an application belongs to group 1, it does not appear initially on the Main Menu. However, users can still add the application to the session profile list or access the application with a START command. (See the *User's Guide*.) Therefore, specifying GROUP=1 does not serve the same purpose as creating an authorized application list. In the APPLDEF commands provided in members KLGCAPLS and KLSCAPLS of DDNAME TLVCMDS, SINGLE applications use GROUP=100; multisession applications use

GROUP=200; and CLSDST PASS applications default to GROUP=0.

HELP	Specifies the help panel displayed when the user selects the application with the H action code.
IMS	CL/GATEWAY for IMS only. This parameter is used when CL/GATEWAY for IMS session services are required for access control in IMS/DC application subsystems. Omit this parameter or specify IMS=NO if you are not using CL/GATEWAY for IMS session services. The default is NO.

INITDLG

Specifies a virtual session initialization dialog for a multisession environment. This dialog receives control immediately after session establishment.

The INITDLG parameter performs the same function in a multisession environment that the LOGON parameter performs for SINGLE sessions. If you specify MULTSESS=NO, you must use LOGON instead of INITDLG.

If you use the INITDLG parameter, you must specify MULTSESS=YES for the application.

Upon entry to *dlgname*, the variable &sysparm will contain the session ID.

LOGON

Defines an initialization sequence to pass to the destination application immediately after session establishment. The sequence primes the application by entering one or more initial transactions. For example, the initialization sequence could complete a logon, which LOGON would pass directly to the application.

You can use LOGON with SINGLE sessions only. Therefore, you must also specify the POOL parameter. If you specify MULTSESS=YES, you must use INITDLG instead of LOGON.

The initialization sequence can include both literal data and variables. For example, you can imbed in it the variables that contain a user ID and password. (See the *Dialog Language Reference Manual*.)

You can also include hexadecimal data. Use a backslash (\) to indicate that the next two characters represent one byte of hexadecimal data. For example, \C1 represents the hexadecimal constant C1.

Three kinds of hexadecimal data are interpreted as “wait for message”:

- **\FF**

  The gateway transmits the data preceding the \FF, then waits for the application to generate a response. When the gateway receives the application's response, it discards the response and then transmits the next string of data from the terminal, up to but excluding the next \FF. When it reaches the end of the initialization sequence, the gateway displays the response returned with the last transaction. If it encounters \FF without any preceding data, or if it encounters two adjacent \FFs, the gateway stops transmitting, and the virtual session waits for an outbound chain from the application.

The \FF is satisfied by any non-null RU.

- **\FD**

  If you imbed \FD in the logon sequence, the gateway transmits the next outbound chain from the application to the terminal (rather than discarding the outbound data). This allows the user to view the outbound chain at the terminal.

- **\FE**

  For read partition query operations, imbed \FE in the logon sequence. The gateway passes the query to the terminal and waits for a response, which it then passes to the application. Logon string processing is suspended until the terminal responds to the message received.

MESSAGE Contains a message (up to 20 characters long) about the application status. A customized dialog can route the message and display it at users' terminals. However, the message is not stored and is associated only with the session defined by the APPLDEF command.

In general, the IMBRCST command is a more efficient method to inform users of changes in application status. See the *Operator's Guide* for more information on the IMBRCST command.

MULTSESS Tells CL/SUPERSESSION to establish a virtual session with a multisession application.

If MULTSESS=YES, the gateway activates a virtual session. If MULTSESS=NO, the gateway establishes a SINGLE session. While this SINGLE session is active, the user cannot use any multisession product features.

If you specify MULTSESS=YES, you must also specify the POOL parameter. The default is MULTSESS=YES.

NEWGROUP Changes the group assignment for an existing application definition. This does not create a new application definition; it simply associates a new group number with the existing application definition.

NOLIST Abbreviates messages that confirm application definitions or updates. If you omit this parameter, the system displays all APPLDEF parameters for each command. NOLIST is particularly useful in the startup CLIST, because it eliminates unnecessary message traffic and log entries.

ORDER	Determines initial placement on the Main Menu. Applications appear on the menu in high-to-low order within each group. By default, all applications are arranged by group number, by order number, and then alphabetically by session ID. If more than one application has the same order number, the system displays the duplicate applications alphabetically.
POOL	Specifies the virtual terminal pool.
	vsmpool — Specifies the virtual terminal pool defined by a VSM command.
	`'*'` — Derives the virtual terminal pool name from the POOL data element.
	If POOL is not specified or if POOL is specified without `'*'` or `'&DEFPOOL'`, the gateway ignores the POOL data element. If the POOL parameter is omitted, the application becomes CLSDST PASS.
PRINTER	Specifies whether to establish the session if the printer is not available. The default is NONE.
	REQUIRED — Specifies that the print node must be resolved before establishing the session.
	OPTIONAL — Establishes the session even if any operation concerning the printer fails (for example, if the printer is busy).
	NONE — Bypasses all printer operations.
PRTPOOL	Specifies the virtual printer terminal pool that establishes sessions between the user and the application.
	vsmpool — Identifies the virtual printer pool defined by a VSM command.
	`'*'` — Derives the virtual printer pool name from the PRTPOOL data element at the time of application access.
	If PRTPOOL is not specified or if PRTPOOL is specified without `'*'`, the gateway ignores the PRTPOOL data element. (Required only for virtual printer sessions.)
REMOVE	Deletes a previously defined APPLDEF application definition. If the original APPLDEF statement specifies a GROUP number, the APPLDEF REMOVE statement must specify that same GROUP number.
	When you specify REMOVE for an existing APPLDEF, and the session ID occurs in one or more authorized application lists, the application is removed from those lists. To re-establish the

application, you must reissue both the APPLDEF command and the corresponding APPLIST command.

SIMLOGON Queues a CLSDST PASS session (no POOL parameter) to the gateway after termination of the session. The SIMLOGON parameter issues a SIMLOGON OPTCD=Q on behalf of the user's terminal logical unit. When the user logs off the session, the session with the gateway is automatically re-established.

If the user selects a CLSDST PASS application, the gateway passes control to the requested application. When the user logs off the application, control returns to VTAM. If the gateway is specified as a controlling application (VTAM LOGAPPL), or if the user selects the gateway from VTAM USS, the user must repeat the logon sequence.

SIMLOGON automatically re-establishes the session with the gateway without implementing a VTAM LOGAPPL. In addition, if the gateway requires that the user re-enter a user ID and password, this information can be passed to the gateway as userdata when the gateway session is re-established. The user sees only a direct return to the gateway when the application terminates.

To activate SIMLOGON, specify either YES or '*data*', where '*data*' is any valid string or string expression to be passed to the controlling gateway as userdata. Userdata must be enclosed in single quotes. If SIMLOGON is to be activated without passing any userdata, specify YES (without single quotes). The default is NO.

Important: You can use either the FORWARD LOGAPPL or the APPLDEF SIMLOGON, but do not use both for the same application.

TERMDLG Specifies a virtual session termination dialog for multisession applications/environments only. This dialog receives control upon detection of the VSSTERM dialog function being issued. The specified dialog cannot contain a VSSTERM function. It will cause the termination dialog to execute twice.

Upon entry to *dlgname*, the variable &sysparm will contain the session ID.

Note: This dialog does not get control when a user performs the normal logoff sequence for an application.

USERDATA Transmits a userdata sequence to the destination application during session establishment. Some applications, including TSO and TSO/E, inspect the contents of userdata and use the data to complete or supplement the logon process.

The userdata string can include both literal data and session variables. For example, you can imbed in the initialization sequence the variables that contain a user ID and password (&VIGUSER and &VIGPSWD).

You can also include hexadecimal data. Use a backslash (\) to indicate that the next two characters represent one byte of hexadecimal data. For example, \C1 represents the hexadecimal constant C1.

The USERDATA parameter takes effect before LOGON or INITDLG.

Specifying More than One APPLDEF per Application

In most cases, there is a one-to-one relationship between network applications and APPLDEF application definitions. However, you might want to specify several APPLDEFs for a single application, to provide unique session controls and services through each access path.

For example, you could define a single IMS/DC application region with two APPLDEFs, the first for users on dedicated lines and the second for dial-in users. The users on dedicated lines can be passed directly to the IMS/DC subsystem. Dial-in users can be routed into a virtual session using a virtual terminal pool, allowing idle sessions to be detected and terminated, thereby increasing resource availability.

You can also use separate APPLDEF commands to define subapplications grouped by function in a single subsystem. For example, payroll, accounts receivable, accounts payable, and order processing can be different applications in the same IMS or CICS subsystem. CL/GATEWAY can list each of these applications separately on the menu. As the user selects the application, CL/GATEWAY can establish the session directly with the subapplication in IMS or CICS. This feature requires SSPL programming with an initial dialog.

Changing Application Definitions Dynamically

You can issue the APPLDEF command from either the CT/Engine operator facility or the MVS console, to override any of the application parameters in the current application definition. For example, to specify a new virtual terminal pool dynamically, you could issue the following command:

```
APPLDEF TSOPROD GROUP=100 POOL=VIRTPOOL
```

VIRTPOOL now replaces the existing virtual terminal pool. This change becomes effective for new logons, but does not affect users already in session with TSO.

To delete a parameter from an APPLDEF command, specify the parameter without a value. For example, to delete the POOL parameter from the APPLDEF command shown above, you would issue this command:

```
APPLDEF TSOPROD GROUP=100 POOL=
```

Note: If the GROUP parameter is used in the original application definition, you must specify the GROUP parameter when you reissue the APPLDEF command.

Using Application Definition Help

To create a site-defined help pop-up for a session ID on the main menu, follow these steps:

1. Create or revise an application definition to include this parameter:

   ```
   HELP=dialog1
   ```

   where *dialog1* is a new member name and is used in the next step. For information about creating an application definition, see the *Operator's Guide*.

2. Create member *dialog1* in the *&rhilev*.RLSPNLS dataset by copying from this member:

   ```
   &thilev.TLSPNLS(KLSH1HLP)
   ```

3. Modify the third line in new member *dialog1*. which reads:

   ```
   DIALOG KLSSHELP KLG010H&vsplang
   ```

   so that it reads:

   ```
   DIALOG KLSSHELP dialog2
   ```

   where *dialog2* is another new member name and is used in the next step.

4. Create member *dialog2* in the *&rhilev.*.RLSPNLS dataset by copying from this member:

   ```
   &thilev.TLSPNLS(KLG010Hn)
   ```

   where *n* is the national language designation (1 = English). For information about language selection, see the *User's Guide*.

5. Modify the text in member *dialog2* to meet your site's help requirements.
6. From the operator interface, refresh *dialog1* and *dialog2*. For information about refreshing dialogs, see the *Operator's Guide*.
7. From the operator interface, activate the APPLDEF for which you defined help. For information about activating an application definition, see the *Operator's Guide*.

Result: Your application definition is activated. To select help, enter the **H** action code next to a session ID on the main menu.

Application Definition Examples

The following examples show how to use some of the APPLDEF parameters.

Example 1: The USERDATA and SIMLOGON Parameters

```
APPLDEF   TSO       DEST(TSO1)                       -
          USERDATA('LOGON &VIGUSER ACCOUNT(&VIGACCT)') -
          SIMLOGON('&VIGUSER &VIGPSWD')
```

In this example, the TSO session ID represents a TSO subsystem. When the user selects TSO, CL/GATEWAY uses the logical unit name specified in the DEST parameter to establish a session with TSO. Since a virtual terminal pool is not specified with the POOL parameter, the system initiates the session with a CLSDST PASS.

The USERDATA parameter defines a character string to pass to the application when CL/GATEWAY establishes the session. To construct the userdata string, CL/GATEWAY substitutes values for all variables in the string. In the example, the &VIGUSER variable contains the user ID, and &VIGACCT contains the account number, so CL/GATEWAY replaces the variables with the proper user ID and account number. When the system passes the session, TSO uses the userdata string to start the logon process.

The SIMLOGON parameter defines a character string to pass back to the controlling gateway when TSO terminates. The &VIGUSER variable contains the user ID, and &VIGPSWD contains the password. When the TSO session ends, the user is automatically logged back onto the gateway. If that gateway resolves the USERID data element by USERDATA(0) and the PASSWORD data element by USERDATA(1), the user need not re-enter a user ID and password.

Example 2: The ALTDEST and MESSAGE Parameters

```
APPLDEF   CICS     DEST=CICSRGN1 ALTDEST=CICSRGN2      -
                   DESC='HOTEL RESERVATION SYSTEM'     -
                   MESSAGE='AVAILABLE THIS WEEKEND'
```

In this example, the CICS session ID can represent either of two CICS application regions: CICSRGN1 and CICSRGN2. When the user selects CICS, the gateway tries to establish a CLSDST PASS session with the primary destination application (CICSRGN1) defined by the DEST parameter. If that application is not available, the gateway next tries to start a CLSDST PASS session with the alternative destination application (CICRGN2) defined by ALTDEST. CICSRGN2 provides a backup for CICSRGN1.

A CT/Engine operator can update the status message dynamically by reissuing the APPLDEF command with a new MESSAGE parameter. The following dialog example, executed by a trigger, allows users to see the current message text (or the end user can issue an I action code on the CL Main Menu for the desired menu item.

```
)option popup level(1)
)declare
appl scope(local)
)prologue
if not &sysparm
 do
 set appl (vssinfo('FOREGRID'))
 viggap(findappl '',&appl)
 end
else
viggap(findappl '',&sysparm)

)attrs
'*' type(output) color(yellow)     display(high)
'<' type(output) color(turquoise)  display(normal)
'>' type(output) color(turquoise)  display(high)

)body top input
* Message text for session id &vigtoken

> SESSION ID        MESSAGE
< ----------        ----------------------------------------
< &vigtoken         &vigmess
)body bottom
*HIT ENTER TO EXIT                   <&sysid
)epilogue
set appl ''
return
```

Example 3: An Initialization Sequence

```
VSM        DEFINE     VIRT3270                            -
                      KLST0001 THROUGH(10)                -
                      LOGMODE(D4A32782)
*
APPLDEF    SYSIMS     DEST=PRODIMS                        -
                      POOL=VIRT3270                       -
                      MULTSESS=NO
                      LOGON='\FF\6D\FF\7D\40\40/FOR SIGNON+
\FF\7D\40\40\11\4E\D6&VIGUSER\11\50\F6&VIGPSWD'
```

This example shows how you can define virtual sessions and programmed initialization sequences. The POOL parameter identifies the virtual terminal pool established by the VSM command. When a user selects SYSIMS, the gateway establishes a SINGLE session, since POOL is specified and MULTSESS=NO.

The LOGMODE parameter defines a virtual 3270 device.

The LOGON parameter defines an initialization sequence. With this sequence, the gateway discards the IMS greeting and obtains a transaction entry MFS panel, using the IMS /FOR command. The gateway then presents the USERID and PASSWORD data elements to IMS in the proper buffer locations, thereby automating the complete logon process to IMS.

```
'\FF\6D\FF\7D\40\40/FOR SIGNON+
\FF\7D\40\40\11\4E\D6&VIGUSER\11\50\F6&VIGPSWD'
```

The hexadecimal FF character is a transmission turnaround indicator. Because no data precedes the \FF, the gateway simply waits for an IMS message. Since more transactions are present in the initialization sequence, the greeting is discarded when received.

The hexadecimal 6D character is the 3270 terminal CLEAR screen order. The \FF marks the end of an intermediate transaction. Thus, the clear key code is sent to IMS/DC, just as if issued by the user. IMS/DC returns a keyboard unlock sequence to the virtual terminal.

The initialization sequence continues by requesting a SIGNON panel with the IMS /FOR command. The hexadecimal 7D4040 segment consists of the 3270 AID code representing the Enter key and the cursor address of row 1 and column 1 of the command line. The /FOR command and panel name follow directly behind these control characters. The hexadecimal FF character instructs the gateway to discard the signon panel when it is received.

Finally, the initialization sequence supplies the user ID and password to IMS in the specified fields of the SIGNON panel. The hexadecimal 7D4040 represents the 3270 AID code for the Enter key and cursor position. The hexadecimal 114ED6 represents the 3270 SBA (set buffer address) code and screen location for row 12 column 39. The session variable &VIGUSER, which contains the USERID, is presented at this location. The hexadecimal 1150F6 represents the 3270 SBA code and screen location for row 14 column 39. The variable &VIGPSWD, which contains the password, is presented at this location. This will appear to IMS as if the user entered a user ID and password, and then pressed Enter. Because this is the last sequence in the LOGON string, the IMS reply to the signon transaction passes directly to the user's terminal.

| **Example 5: CICS Logoff Sequence**

| A sample CICS logoff sequence is included in member KLSTERMD of
| *&thilev.*TLSSAMP. The example shows how to perform a normal CICS SIGNOFF
| instead of a forced logoff for CICS.

Creating Authorized Application Lists

CL/GATEWAY provides *authorized application lists* to help you restrict user access to applications.

A *static* application list is an application list you define *in* CL/GATEWAY, by invoking the APPLDEF and APPLIST commands. Once an application list is defined, it does not change unless you redefine it or remove the applications from the system.

A *dynamic* application list is an application list defined *outside* CL/GATEWAY by an external security system. The list is created dynamically when a user logs onto CL/SUPERSESSION or CL/GATEWAY. The Network Access Manager (NAM) interacts with the security system to validate application access.

The *Basic Configuration Guide* explains how to set up both static and dynamic application lists.

APPLIST Command

The APPLIST command activates or refreshes authorized application list definitions. When you create an APPLIST member or update an existing member, for example, *&rhilev*.RLSPARM(KLGIAPL1), you must issue the APPLIST command to invoke the new list. You must include the APPLIST in the GATEWAY configuration member.

You can also use the APPLIST command to associate a broadcast message group with an authorized application list. This method is useful when the application list is associated with a user or group of users with common network notification requirements.

Any CT/Engine operator can issue the APPLIST command. However, to ensure that the APPLIST commands are active when needed, use a CLIST such as *&rhilev*.RLSCMDS(KLGCAPLT) to invoke the initial APPLIST commands that establish the production CL/GATEWAY environment.

Before you issue the APPLIST command, you must issue an APPLDEF command for an application named in the authorized application list. If you remove and then re-establish an APPLDEF statement, you must reissue an APPLIST command for every application list that includes the application defined in the APPLDEF command.

For detailed information on the APPLIST command, see the *Operator's Guide*.

Example: Creating Application Lists with More than One Group

You may want to create authorized application lists composed of applications whose APPLDEF commands specify different groups. To do so, follow these steps.

1. Create a member in *&rhilev*.RLSPARM called APLSTALL. In APLSTALL enter

   ```
   GROUP=0
   /
   GROUP=100
   /
   GROUP=200
   /
   ```

2. Issue the APPLIST commands as follows:

   ```
   APPLIST APLSTALL ID=G100 GROUP=100
   APPLIST APLSTALL ID=G200 GROUP=200
   ```

3. Add the Applist ID in the user profile to point to G100 for all users you want to have access to Group 100 applications. Add the Applist ID in the user profile to point to G200 for all users you want to have access to Group 200 applications.

Customizing the Network Broadcast Facility

A set of broadcast messages intended for a specific portion of the user community is referred to as a *broadcast group* or BCGROUP. As supplied by Candle, CL/GATEWAY provides the following implementation of broadcast groups:

1. *&rhilev*.RLSCMDS(KLGCBCGP) contains the message commands for each broadcast group.

   ```
   BCGROUP GROUP1 1 'text'
   BCGROUP GROUP1 2 'text'
   BCGROUP GROUP1 3 'text'
   BCGROUP GROUP1 4 'text'
   BCGROUP GROUP1 5 ' '
   BCGROUP GROUP1 6 'text'

   BCGROUP GROUP2 1 'text'
   BCGROUP GROUP2 2 'text'
   BCGROUP GROUP2 3 'text'
   BCGROUP GROUP2 4 'text'
   BCGROUP GROUP2 5 ' '
   BCGROUP GROUP2 6 'text'
   ```

2. *&rhilev*.RLSCMDS(KLGCAPLT) contains the APPLIST commands that create authorized application lists.

   ```
   APPLIST KLGIAPL1 BCGROUP(GROUP1)
   APPLIST KLGIAPL2 BCGROUP(GROUP2)
   ```

   If a user is associated with an authorized application list that specifies BCGROUP, the user sees the messages for that broadcast group after logging onto a gateway.

3. *&rhilev*.RLSCMDS(KLGCHGGW) contains the HOSTGATE commands that define CL/GATEWAY entry points.

   ```
   HOSTGATE KLSGW001 ... BCGROUP=GROUP1
   HOSTGATE KLSGW002 ... BCGROUP=GROUP2
   HOSTGATE KLSGW003 ... BCGROUP=GROUP2
   ```

   When a user enters a gateway, the messages for the broadcast group specified in the HOSTGATE command are displayed until overridden by an APPLIST BCGROUP.

4. RLSPARM(KLGINGWY) contains the CL/GATEWAY startup parameters.

   ```
   BCGROUP(GROUP2) -
   MENUSIZE(5)     -
   MONITOR(1:00)
   ```

   The broadcast group specified in KLGINGWY is the global, or default, broadcast group for CL/GATEWAY.

The APPLIST BCGROUP overrides the HOSTGATE BCGROUP, which overrides the KLGINGWY BCGROUP.

A user might see the messages from more than one BCGROUP specification. For example, if both a global BCGROUP and an APPLIST BCGROUP are specified, but no HOSTGATE BCGROUP is specified, the user sees the global BCGROUP messages at entry validation. Once logged onto a gateway and associated with an application list, the user sees the APPLIST BCGROUP messages.

Whenever a broadcast group is updated or added, panel KLSBULL1 is requested for all users, even if they are associated with a different broadcast group.

Creating a Broadcast Group

The BCGROUP command can be issued by a CT/Engine or MVS operator to define, update, or delete a message in a broadcast group. A new broadcast group is created whenever a BCGROUP command adds a message to a previously unknown broadcast group. Similarly, a broadcast group is removed from the system when the last message it contains is deleted.

Candle recommends that you define your broadcast groups in a CLIST, such as *&rhilev*.RLSCMDS(KLGCBCGP). BCGROUP commands issued subsequently by an operator update the messages but do not affect the contents of the CLIST.

Here is the format of the BCGROUP command:

```
BCGROUP bcgroup line  ['message']
```

bcgroup	The name of the specific broadcast group to be defined or updated. Broadcast group names cannot exceed 8 characters.
line	A decimal digit, from 1 through 6, that identifies the message line affected. The action of the BCGROUP command upon the broadcast group is determined by this operand. If the message line does not currently exist, it is added. If the message line exists, it is replaced with the text provided. If no text is provided, the message line is deleted.
message	A valid string expression, however, quotes, apostrophes, and hex characters are not allowed. Messages that do not fit on a display line of the user's terminal are truncated.

When you change the messages in KLGCBCGP, you must initialize the changes by issuing the KLGCBCGP command from either the CT/Engine operator facility or the MVS console.

BCGROUP Command Example

In the example below, a broadcast group named IMSUSERS is defined. Lines 1 and 2 are assigned initial messages. All other lines remain null by default.

```
BCGROUP  IMSUSERS  1  'IMS WILL BE UP THIS WEEKEND'
BCGROUP  IMSUSERS  2  'HOWEVER, IT IS DOWN RIGHT NOW!'
```

At a later point, an operator can delete message line 2 from the IMSUSERS broadcast group as shown in the following example.

```
   BCGROUP  IMSUSERS  2
```

Customizing the News Facility

You can use the CL/GATEWAY news facility to provide network news items. Users can access the news items by issuing the NEWS command from the Main Menu.

Dialog KLSNEWSn (where *n* is a one-digit national language identifier; for example, 1 for English) is a sample panel for the news facility. As supplied, it lets users select from three news topics:

1. CL/GATEWAY services
2. Terminal pooling
3. CL/GATEWAY session services for IMS/DC

Text for the three topics is provided in *&rhilev.*RLSPNLS members KLSNEW1*n*, KLSNEW2*n*, and KLSNEW3*n*, respectively. You can customize the news facility by modifying those dialogs and adding new dialogs.

If you change the dialogs, use the CT/Engine operator interface REFRESH panel command. That way, when the NEWS command is issued, your most recent text changes are displayed.

Chapter 6. Additional Customization for IMS

This chapter explains the CL/GATEWAY for IMS configuration and tells how to customize CL/GATEWAY for IMS. For basic configuration instructions, see the *Basic Configuration Guide*.

CL/GATEWAY for IMS provides session establishment and session integrity services specifically designed for IMS/DC environments.

Virtual sessions and virtual terminal pools are particularly attractive in an IMS/DC environment. Virtual sessions can eliminate IMS/DC system regenerations and provide virtual storage constraint relief.

When CL/GATEWAY for IMS manages access to an IMS/DC subsystem, the CL/GATEWAY for IMS session services can ensure the privacy of messages presented to a user through a serially shareable IMS/DC physical connection (PTERM). You can select the session service most appropriate for your IMS/DC environment. CL/GATEWAY for IMS performs session services immediately before establishing the new session. All processing is transparent to the user.

- The *session cleanup* service uses a PTERM-based approach to identify and eliminate residual messages queued to CRTs and printer terminals connected by shared communications paths. The CL/GATEWAY for IMS *DEQUEUE* option activates session cleanup.
- The *LTERM assignment* service manages display and printer logical terminals (LTERMs) associated with individual users or groups of users. LTERM assignment controls message delivery on shared communications paths. The CL/GATEWAY for IMS *ASSIGN* option activates LTERM assignment.

Customizing CL/GATEWAY for IMS

To customize CL/GATEWAY for IMS for your site, complete the following steps.

Defining the Control Session

If you plan to have users log onto IMS using CL/GATEWAY for IMS through CL/SUPERSESSION, perform steps 1 and 2. If you plan to have users log onto IMS only through CL/SUPERSESSION, perform step 2.

Note: CL/GATEWAY for IMS services are available only if you are a licensed CL/GATEWAY for IMS user.

Step 1. Copy

&thilev.TLSCMDS to *&rhilev*.RLSCMDS.

Step 2. Customize the IMS configuration definition.

Member KLICIDEF in *&rhilev*.RLSCMDS is used to define IMS/DC applications when the IMS parameter of the APPLDEF command requests CL/GATEWAY for IMS session services (ASSIGN and/or DEQUEUE).

The IMS command identifies the

- VTAM application logical unit name for IMS
- name of the virtual MTO pool
- various characteristics of the IMS/DC subsystem

Globally change IMS1 in *&rhilev*.RLSCMDS(KLICIDEF) to your VTAM network name assigned to IMS.

Step 3. Include definitions in the IMS gen.

Member *&rhilev*.RLSSAMP(KLGIMSGN) contains the terminal definitions used in the sample configurations for CL/GATEWAY for IMS. Do not make any changes at this time.

1. Copy *&rhilev*.RLSSAMP(KLGIMSGN) into your IMS gen.
2. Assemble the IMS gen.

Defining Selections

CL/GATEWAY for IMS defines the IMS/DC applications with the APPLDEF commands found in the KLSCAPLS member of *&rhilev*.RLSCMDS. The IMS parameter of the APPLDEF command specifies which CL/GATEWAY for IMS session services you want to use: ASSIGN, DEQUEUE, or both ASSIGN and DEQUEUE. When you use the IMS parameter, the DEST parameter points to an IMS command, *not* to your VTAM network name for IMS.

Assigning Logical Terminals to Users

When a user logs onto an IMS application through CL/GATEWAY for IMS with the ASSIGN option, CL/GATEWAY for IMS assigns the user ID as the logical terminal (LTERM) ID by default. The LTERM assignment can be easily matched with user needs by customizing the data sources used for the LTERM element.

The following examples show several methods of assigning logical terminals to users. Member KLSLTASS of *&thilev*.TLSSAMP also contains an LTERM assignment example.

Assigning an LTERM with NAM:

1. Log onto APPLID KLGICFG2 or KLGICFG3.
2. Issue the following command through the CT/Engine operator facility:

   ```
   NAM SET userid LTERM:USER001
   ```

3. Modify member KLGICFG1 (or whatever member you are using for the gateway configuration) in *&rhilev*.RLSPARM from

   ```
   LTERM                         -
         USERDATA(LTERM)         -
         PANEL(KLGDIMS)          -
         PROMPT('ENTER LTERM')   -
         HELP(KLGWLTRM)          -
         DEFAULT(&VIGUSER)
   ```

   to

   ```
   LTERM                         -
         NAM
   ```

4. To initialize the change, issue the following commands through the CT/Engine operator facility:

   ```
   CLOSE KLSGW001
   KLGCHGGW
   ```

Assigning an LTERM with the Default Data Source

1. Modify member *&rhilev*.RLSPARM(KLGICFG1) from

   ```
   LTERM                         -
         USERDATA(LTERM)         -
         PANEL(KLGDIMS)          -
         PROMPT('ENTER LTERM')   -
         HELP(KLGWLTRM)          -
         DEFAULT(&VIGUSER)
   ```

   to

   ```
   LTERM                                     -
         DEFAULT('L&SYSEDIT((0,7)&SYSTERM)')
   ```

   In this example, the LTERM assigned is the terminal ID prefixed by the letter L and truncated after the seventh character.

   The default data source can be a terminal ID (&SYSTERM), user ID (&VIGUSER), or any system or user variable.

2. To initialize the change, issue the following commands through the CT/Engine operator facility:

```
CLOSE KLSGW001
KLGCHGGW
```

Assigning an LTERM through a Dialog in

&rhilev.RLSPNLS(KLGDIMS)

1. Modify the dialog in KLGDIMS as follows:

   ```
   SET VIGLTERM 'L&SUBSTR(&SYSTERM,0,7)'
   ```

   In this example, the LTERM assigned is the terminal ID prefixed by the letter L and truncated after the seventh character. A terminal ID (&SYSTERM), user ID (&VIGUSER), or any system or user variable can set VIGLTERM.
2. To initialize the dialog, issue the following command through the CT/Engine operator facility:

   ```
   REFRESH PANEL KLGDIMS
   ```

Note: In all cases, the derived LTERM must be defined in the IMS gen. LTERMs *USER001* through *USER005* are included in CL/GATEWAY for IMS in *&rhilev*.RLSPNLS(KLGDIMS).

CL/GATEWAY for IMS Session Services

At many sites, a number of users share a pool of PTERMs using a variety of port contention techniques. Periodically, these users receive messages from transactions scheduled by previous users. This can occur for a number of reasons, the most common being a premature terminal disconnection on a dial-in line.

When a terminal's connection breaks, messages generated during its transaction remain in the message queue until the session on that PTERM is re-established. However, dynamic selection of a communications path in the network may cause the user to reconnect on a different PTERM and LTERM. Consequently, the queued messages are not delivered. When the original PTERM is later assigned to another user, those messages are presented to the wrong user. The session cleanup and LTERM assignment services of CL/GATEWAY for IMS eliminate this problem.

Session Cleanup

The *session cleanup*, or *DEQUEUE*, service purges residual data from the message queue. Only messages destined for the PTERM selected for the new session are deleted. Because cleanup occurs at the PTERM level, it requires no specific LTERM designations.

Additionally, when a session is prematurely broken during a conversational transaction, the PTERM may be left in response mode. Session cleanup terminates the transaction, further ensuring the integrity of each new session.

You can extend session cleanup to include a printer PTERM associated with the terminal PTERM.

Logical Terminal Assignment

If you rely on LTERM-oriented transaction and message security, *LTERM assignment* can extend security to shared PTERMs. CL/GATEWAY for IMS selects an LTERM name associated with the user, or derives the LTERM name from an algorithm or exit routine you specify.

CL/GATEWAY for IMS reassigns any LTERMs previously associated with the shared PTERM, to ensure that messages currently queued or produced by transactions waiting to run are retained for delivery to the proper user. Next, the gateway acquires a specific LTERM name associated directly with the user, and assigns it to the PTERM. When the new session is established, the user can recover any messages previously queued for that LTERM.

If a printer needs to be associated with this terminal (prtnode data element in the configuration), then an LTERM can be assigned to the printer through the PRTLTERM data element of the configuration.

Session Service Implementation: Virtual MTOs

A programmed Master Terminal Operator interface, referred to as the *virtual MTO*, implements session services. CL/GATEWAY for IMS issues IMS operator commands to display the current status of the real or virtual terminal PTERM associated with the user. Depending on the results and the particular session service selected, the virtual MTO issues additional commands to start and stop the PTERM, purge queued messages, or make LTERM assignments.

The virtual MTO appears to IMS as an authorized operator using standard IMS/DC command facilities. IMS provides the ability to limit operator command authorities. The programmed MTO requires the ability to issue the commands shown below. If password protection is generated for any of these commands, the appropriate password can be supplied in the IMS statement that defines the IMS/DC environment.

```
ASSIGN      START     RSTART     EXIT
DISPLAY     STOP      DEQUEUE
```

To activate the virtual MTO interface, you must establish one or more sessions between CL/GATEWAY for IMS and IMS/DC. Hardware and IMS/DC software modifications are not required. To establish the communication path between the virtual MTO and IMS/DC, you need to define VTAM, CL/GATEWAY for IMS, and IMS resources.

The same set of virtual MTOs can be used for more than one IMS subsystem. See "Virtual Terminal Sharing" on page 46.

VTAM Requirements

Use a VTAM APPL statement to represent each virtual MTO. The virtual MTO consists solely of program logic residing in CL/GATEWAY for IMS, but it appears to IMS as a physical 3767 (LU1) terminal.

For best performance, the number of virtual MTOs should approximate the number of concurrent IMS logons processed by the gateway during an average one-second period. Two to four sessions usually suffice, even in large IMS installations.

Each IMS subsystem perceives the virtual MTO as a single terminal device, although the application logical unit can maintain many concurrent sessions.

Specify the AUTH parameter as illustrated in the example below. Its subparameters convey the following capabilities:

ACQ	Allows CL/GATEWAY for IMS to initiate the virtual MTO session with the IMS/DC application subsystem.
NVPACE	Deactivates VTAM message traffic pacing on this path. NVPACE is optional; it is included for performance purposes only.

When you define more than one virtual MTO APPL, their network names should consist of a constant alphanumeric prefix followed by a numeric suffix. In the example below, the network names have the form KLIVMTOn, where KLIVMTO is the prefix and n represents the decimal suffix.

```
KLIVMTO1  APPL  AUTH=(ACQ,NVPACE)  IMS/DC VIRTUAL MTO#1
KLIVMTO2  APPL  AUTH=(ACQ,NVPACE)  IMS/DC VIRTUAL MTO#2
KLIVMTO3  APPL  AUTH=(ACQ,NVPACE)  IMS/DC VIRTUAL MTO#3
KLIVMTO4  APPL  AUTH=(ACQ,NVPACE)  IMS/DC VIRTUAL MTO#4
```

IMS Requirements

The IMS/DC terminal definition must include a PTERM and associated LTERM definition for each virtual MTO terminal. The CL/GATEWAY for IMS virtual MTO program logic emulates the 3767 (LU1) protocol.

The following example depicts IMS gen definitions for a virtual MTO pool containing two terminals. For additional IMS gen definition examples, see *&rhilev.*RLSPNLS(KLSPV3M).

```
TYPE UNITYPE=3767,FEAT=IGNORE,              X
     OPTIONS=(TRANRESP,NOMFS,OPNDST,BSELM), X
     MSGDEL=NONIOPCB
*
     TERMINAL NAME=KLIVMTO1  PTERM FOR VIRTUAL MTO SESSION
```

```
NAME     VMTO1          LTERM FOR VIRTUAL MTO SESSION
TERMINAL NAME=KLIVMTO2  PTERM FOR VIRTUAL MTO SESSION
NAME     VMTO2          LTERM FOR VIRTUAL MTO SESSION
```

An IMS/DC terminal operates either in response mode, waiting for a reply before unlocking the keyboard, or in nonresponse mode.

- If the terminal is in non-response mode and if response time is slow, the terminal can be unlocked prematurely. This allows the user to enter more data before receiving a reply from IMS/DC to the previous data entry. The second data entry will be lost.
- If the **TRANRESP** option is coded, and the transaction code definition indicates non-response mode, the keyboard could be unlocked prematurely.

The **FORCRESP** option specifies response mode for all transactions entered at the terminal. This option should be considered if response time is a problem.

You can establish virtual sessions between the end user and IMS/DC. If you do so, the IMS/DC terminal definition must include a PTERM for each CL/GATEWAY virtual terminal node. PTERMs supporting virtual sessions with the end user should reflect the actual device types used. If CL/SUPERSESSION is installed, IMS/DC can be defined as a multisession application.

CL/GATEWAY for IMS Requirements

Use the VSM DEFINE command to define the virtual MTO terminal pool to CL/GATEWAY for IMS. Include LOGMODE=SCS to designate a VTAM bind image appropriate for the 3767 (LU1) interface.

For CL/GATEWAY functions (such as DEQUEUE/ASSIGN) to operate correctly, the IMS application must be defined in APPLDEF. The following VSM DEFINE commands define two virtual terminal pools.

```
VSM DEFINE  IMSOP    KLSTMTO1  THROUGH(4)     LOGMODE(SCS)
VSM DEFINE  SINGLE   KLST0001  THROUGH(0004)  LOGMODE(D4A32782)
```

The first VSM command defines a pool, named IMSOP, of virtual MTOs. The second VSM command defines a pool, named SINGLE, of PTERMs for SINGLE sessions.

For a complete explanation of the VSM DEFINE command, see the *Operator's Guide*.

The IMS command that defines the IMS/DC subsystem must include a POOL operand to specify the virtual MTO pool created by the VSM DEFINE command.

Defining an IMS/DC Subsystem

The IMS command

- specifies the destination pointed to by the DEST parameter of an APPLDEF command for IMS
- identifies the network name for IMS
- identifies the name of the virtual master terminal operator (MTO) pool
- identifies the "dummy" virtual terminal to which LTERMs are assigned when they are disconnected

  For information on the dummy PTERM, see "LTERM Assignment: ASSIGN" on page 113.

Place IMS commands in the command library *&rhilev*.RLSCMDS) to ensure that the CL/GATEWAY for IMS application definitions are available at system startup time.

The IMS command and the associated VSM and APPLDEF commands shown here illustrate the operands used to define a typical gateway for IMS that includes the LTERM assignment session service.

```
IMS       IMSASSIG     APPLID(PRODIMS)          -
                       POOL(KLGTMTO)            -
                       DUMMY(FAKENODE)          -
                       MIN(1) MAX(4)
*
VSM       DEFINE       KLGTMTO                  -
                       KLIVMTO1 THROUGH(4)      -
                       LOGMODE(SCS)
*
APPLDEF   IMS          DEST(IMSASSIG)           -
                       IMS(ASSIGN)
```

For detailed information on the IMS command, see the *Operator's Guide*.

Session Cleanup: DEQUEUE

To activate session cleanup, include IMS=DEQUEUE in the APPLDEF command that defines an IMS/DC application. The DEST operand of the APPLDEF command specifies the IMS command that defines the IMS/DC subsystem. This example shows an APPLDEF command for a SINGLE IMS application.

```
APPLDEF    IMSDEQ                              -
           IMS=DEQUEUE                         -
           DEST=IDEQUEUE                       -
           DESC='IMS DEQUEUE (SINGLE)'         -
           POOL='&DEFPOOL'                     -
           MULTSESS=NO                         -
           HELP=KLIPGD1                        -
```

```
            GROUP=100
```

For a complete explanation of the APPLDEF command, see “CL/SUPERSESSION and CL/GATEWAY Customization” on page 65.

DEQUEUE Virtual MTO Command Sequence

If you specify IMS=DEQUEUE in the APPLDEF command, the programmed virtual MTO issues a series of IMS operator commands to inspect the status of the PTERM to be used in establishing a new session. It then issues additional commands as needed, to purge any leftover messages and reset the PTERM.

/DIS CONV HELD NODE pterm
The output produced by the DISPLAY command identifies conversations against the selected PTERM.

/STO NODE pterm
Stop the PTERM.

/EXI CONVERSATION conv_id NODE
Exit conversation *conv_id* for node PTERM.

/STA NODE pterm
Start the PTERM.

This STOP, EXIT, and START command sequence resets the response mode and terminates any conversational transactions in progress.

/STO NODE pterm
Stop the PTERM.

/DEQ NODE pterm PURGE
Dequeue messages in the PTERM.

/STA NODE PTERM
Start the PTERM.

This command sequence deletes all messages queued to the LTERM currently assigned to the PTERM. If a command issued during session cleanup fails, the IMS/DC error messages are listed in the VIEWLOG database, and the session is not established. The Main Menu informs the user of the error.

LTERM Assignment: ASSIGN

To activate LTERM assignment, include IMS=ASSIGN in the APPLDEF command that defines an IMS/DC application. The DEST operand of the APPLDEF command specifies the IMS command that defines the IMS/DC subsystem. The following example shows an APPLDEF command for a SINGLE IMS application.

```
   APPLDEF    IMSASSGN                           -
              IMS=ASSIGN                         -
              DEST=IASSIGN                       -
              DESC='IMS ASSIGN (SINGLE)'         -
              POOL='&DEFPOOL'                    -
              MULTSESS=NO                        -
              HELP=KLIPGD1                       -
              GROUP=100                          -
              LOGON='\FF\6D\FF\7D\40\40/FOR SCREEN'
```

For a complete explanation of the APPLDEF command, see "CL/SUPERSESSION and CL/GATEWAY Customization" on page 65.

If you specify IMS=ASSIGN in the APPLDEF command, the programmed virtual MTO issues a series of IMS operator commands to inspect

- the status of the selected PTERM used to establish the new session for the current user
- the status of the residual LTERMs assigned to that PTERM

The operator command sequence then performs the following steps:

1. Reassigns the currently active LTERMs to the dummy PTERM established for this purpose in the IMS definition.
2. Assigns the LTERM associated with the current user to the selected PTERM.

For more information on virtual MTOs, LTERMs, and PTERMs, see "Session Service Implementation: Virtual MTOs" on page 109 and "IMS Requirements" on page 110.

The LTERM data element must be included in the gateway configuration member. If a printer LTERM is associated with the session, the PRTNODE and PRTLTERM data elements are also required. If you specify PRTNODE and PRTLTERM, the virtual MTO issues the command sequences twice: once for the terminal and once for the printer node. For more information about CL/GATEWAY data elements, see "CL/SUPERSESSION and CL/GATEWAY Customization" on page 65.

If more than one user is assigned the same LTERM, or if a user attempts to log onto IMS from two separate sessions, an LTERM can be reassigned during an IMS session. By using extended LTERM verification (XLV), you can prevent this situation. If you specify XLV in the IMS command, a user will not be allowed access to IMS with an LTERM currently in use. The LTERM assignment command sequence issued by the virtual MTO uses a reserved PTERM to manage inactive LTERMs. When a new session begins on a given PTERM, all LTERMs currently assigned to it are reassigned to this reserved PTERM, known as the *dummy PTERM* (and commonly named FAKENODE).

The dummy PTERM and the LTERM name assigned to each user in the LTERM data element must be identified in the IMS terminal definition. The example below shows system generation statements that define a dummy PTERM and associated LTERMs. For additional IMS gen definition examples, see *&thilev.*TLSSAMP(KLGGWIMS).

```
TYPE  UNITYPE=SLUTYPE2,MODEL=2,                X
      FEAT=(PFK,NOCD,NOPEN),                   X
      OPTIONS=(TRANRESP,COPY,PAGDEL,OPNDST)
TERMINAL  NAME=FAKENODE          DUMMY PTERM
   NAME   USER001                USER LTERM #1
   NAME   USER002                USER LTERM #2
```

ASSIGN Virtual MTO Command Sequence

/DIS ASMT LT lterm
(XLV only) If extended LTERM verification is in effect, the virtual MTO issues this command for the LTERM name contained in the LTERM or PRTLTERM data element. The output produced by the DISPLAY command identifies the PTERM currently assigned to the specified LTERM.

/DIS NODE pterm
(XLV only) The virtual MTO requests the status of the PTERM identified in the previous DISPLAY command. The output from this DISPLAY command determines whether the MTO command sequence continues. If the PTERM is already active, the MTO command sequence halts, and a message appears on the user's terminal if a message field or panel is available.

/DIS CONV HELD NODE pterm
The output produced by the DISPLAY command identifies conversations against the selected PTERM.

/DIS ASMT NODE pterm
The output produced by the DISPLAY command identifies all LTERMs currently assigned to the selected PTERM.

/ASS LT lterm NODE dummy_pterm
The virtual MTO issues an ASSIGN command for each LTERM identified by the DISPLAY command. Each LTERM identified is assigned to the dummy PTERM. Any messages queued for these LTERMs are retained in the message queue. Upon completion of the ASSIGN sequence, the PTERM on which the new session is established has no associated LTERMs.

/ASS LT lterm NODE pterm
This command ASSIGNs to the selected PTERM the LTERM name contained in the LTERM or PRTLTERM data element. Upon completion, the PTERM is associated only with the user's LTERM.

Occasionally, a session prematurely terminated during a conversational transaction prevents reassignment of the LTERM for a new session. If a command issued during the LTERM assignment sequence fails, the IMS/DC error messages are listed in the VIEWLOG database, and an error recovery procedure is invoked. The error recovery procedure issues STOP, START, and RESTART commands that terminate the transaction, eliminate the response mode condition, and restart the assignment sequence.

Combining Session Cleanup and Logical Terminal Assignment

You can activate both session cleanup and LTERM assignment for the same IMS application. To do so, specify IMS='ASSIGN,DEQUEUE' in the APPLDEF command that defines the IMS application.

Specifying both ASSIGN and DEQUEUE can be valuable in an environment that provides LTERM assignment for a subset of the user community, with all users sharing a pool of PTERMs. For the users requiring LTERM assignment, you may need to preserve queued messages so they can be retrieved at a later time.

A user without LTERM assignment receives the LTERM currently assigned to the PTERM associated with the user's terminal. This PTERM may have been in use by a user with LTERM assignment. In this case, the PTERM should be reassigned a default LTERM specification so as to retain the previous user's session. A specification of 'ASSIGN,DEQUEUE' satisfies all users in this situation.

Chapter 7. Additional Customization for CL/SUPERSESSION

You can customize CL/SUPERSESSION by

- executing administrator functions from the Main Menu action bar
- creating new dialogs
- changing dialogs provided by Candle

The *Basic Configuration Guide* gives instructions for executing the administrator functions. This chapter offers suggestions for

- customizing triggers
- using CL/SUPERSESSION with vector graphics
- using CL/SUPERSESSION with the IDMS online mapping facility
- using implicit partitions query reply
- using CL/SUPERSESSION's interface for terminal compatibility
- using data compression
- using CL/SUPERSESSION to implement code point translations
- customizing the CL/SUPERSESSION initialization parameters

Customizing Triggers

The following default trigger dialogs are available. You can change, delete, and add triggers by executing administrator functions from the action bar of the Main Menu.

Note: See "Performance Tips" on page 229 for additional information on adding triggers.

Table 7 (Page 1 of 2). Triggers Supplied with CL/SUPERSESSION

Hot Phrase	Parameter	Hot Key	Action	Dialog
\g	session ID	Enter	Go to the session named. For example, \gtso goes to TSO.	KLSGOTO
\j	window ID	Enter	Jump to the next window or to the window named. For example, \j goes to the next window, and \ja goes to window A.	KLSNEXTW
\l		Enter	Lock the physical terminal.	KLSLOCK
\m		Enter	Display the Main Menu.	none
\n		Enter	Move to the next active session according to the order the sessions were started.	KLSNEXTS
\o	session ID	Enter	Display the virtual terminal options panel.	KLSVTOPT

Table 7 (Page 2 of 2). Triggers Supplied with CL/SUPERSESSION				
Hot Phrase	**Parameter**	**Hot Key**	**Action**	**Dialog**
\p		Enter	Move to the previous active session according to the order the sessions were started.	KLSPREVS
\q	X or N	Enter	Log off CL/SUPERSESSION with X to terminate sessions or with N to exit without terminating sessions.	KLSQUIT
\z		Enter	Toggle the window between zoomed and unzoomed states.	KLSZOOM
@p	session ID	Enter	Print the screen of the foreground session. Specify a session ID to print the screen of a background session. (The action code P performs the same function.)	KLSPRINT

When a trigger is defined as a single key (for example, a function key), the trigger parameter cannot be typed in dynamically. However, when a trigger is defined with a phrase (for example, \g), the trigger parameter can be typed in dynamically from an application.

After a trigger is executed, the cursor always returns to the position it was in before the trigger was executed.

Window Control Trigger Dialogs

CL/SUPERSESSION's trigger dialogs offer an alternative method for performing the following frequently used operations:

- zooming and unzooming a windowed display
- changing the active window

Triggers allow these operations to be performed in one host-interrupt sequence, instead of the control-key followed by function-key method.

The ZOOM dialog accepts no parameter. It simply zooms or unzooms the windowed display in one host-interrupt sequence.

The jump operation lets users specify the *window identifier* of the desired active window. The NEXTWIND dialog executed without a parameter activates the next window in a manner identical to the window control jump function, in one host-interrupt sequence. The NEXTWIND dialog also accepts a trigger parameter that specifies the alphabetic window identifier of the window to be activated. This is similar in concept to the window control cursor positioning method for selecting the new active window. The jump can be performed in a single host-interrupt sequence, and users can jump directly from a zoomed window to a currently invisible window.

Note: Triggers are recognized only if they are entered from a virtual session's display screen. Triggers entered on a dialog panel are passed as input to the dialog.

For details of window control operations, see the *User's Guide*.

File Transfer Trigger

Certain PC-based file transfer applications may work improperly with compression. These applications expect specific hex patterns in a specific order and cannot interpret a compressed 3270 datastream.

You can prevent file transfer problems by adding the \ft trigger, which calls dialog FXFER for the specified session and

- sets outbound data compression to No
- sets inbound data compression to No
- sets query SINGLE to Yes
- suspends timeout

The \ft trigger also inhibits immediate broadcast and sets

- READ-modified for PA keys to No
- READ-modified for ATTN keys to No
- Fullread mode to No

The \ft trigger is a toggle; after file transfer is complete, the user can reissue \ft to reactivate data compression. For information about adding the \ft trigger, see the *User's Guide*.

Vector Graphics

CL/SUPERSESSION is not able to save vector graphics data streams. Therefore, if the physical terminal locks during a session in which vector graphics are displayed, the vector graphics are not redisplayed when the user returns and unlocks the terminal.

When you use vector graphics orders, the screen buffer does not refresh after the VSSFOREG command is issued. However, some applications provide keys that refresh the screen; for example, TSO and VM use PA2.

IDMS Online Mapping

When using CL/SUPERSESSION with IDMS online mapping, it is necessary to turn both inbound and outbound compression off. The read buffer mode must be YES. Use the VSSOPT function in the initial dialog, code your own trigger dialog to issue VSSOPT for IDMS, or use the \o trigger to turn off compression and set the read buffer mode.

Implicit Partitions Query Reply

CL/SUPERSESSION manages the implicit partitions query reply on virtual sessions so that applications receive query reply information that reflects the bind image used for the virtual session.

For example, a terminal user logging on with a Model 4 device (normally the bind image specifies a 24×80 default and a 43×80 alternate screen size) may establish a virtual session where the virtual terminal uses a Model 2 logmode (a 24×80 default and 24×80 alternate screen size). If the application issues a read partition query on the virtual session, CL/SUPERSESSION obtains query data from the physical device and alters the screen size information in the implicit partitions query reply to indicate a 24×80 default and 24×80 alternate screen size before passing the query data into the application.

Terminal Compatibility

CL/SUPERSESSION's interface to the physical terminal allows any virtual session to be mapped into the physical terminal's display space. To view virtual session screen images larger than the physical terminal screen, the user must invoke window control and scroll through the image segments.

You can use administrator functions or dialog variables to control the presentation space size for full-screen panels. See the *Basic Configuration Guide* and the *Dialog Language Reference Manual*.

Outbound Compression

CL/SUPERSESSION can either use or bypass outbound data compression. When compression is turned off, the outbound datastreams from the application to the virtual terminal are subjected to minimal processing by CL/SUPERSESSION before being sent directly to the physical terminal.

When outbound data compression is in effect, CL/SUPERSESSION uses the outbound datastream from the application to update the virtual device buffer that represents the screen image created by the application. The outbound data compression routines then compare this image with the actual contents of the physical display screen. These routines create and transmit the smallest possible datastream that will update the physical display screen to reflect the screen image that was created by the application.

Therefore, you can turn off compression to keep CPU usage to a minimum; or you can use compression to reduce the number of bytes transmitted through the network.

Here are some basic characteristics of outbound data compression:

- Outbound compression can be turned off only when there is no more than one window in use.

  When more than one primary window is created (for example, when the user invokes the window control SPLIT function), CL/SUPERSESSION automatically begins operation with outbound data compression. Therefore, even if an application is defined with the outbound data compression option set to No, this setting is ignored when the user accesses the application through a windowed display. This is true whether the window is zoomed or unzoomed.

- Outbound compression can be turned off only when the screen characteristics of the virtual session match the screen characteristics of the physical device session.

  Since the outbound datastream from the application is sent unchanged to the physical device when compression is turned off, CL/SUPERSESSION ensures that the physical device is capable of following the default and alternate screen size changes expected by the application. If the screen characteristics do not exactly match, CL/SUPERSESSION operates with outbound data compression for that session.

  Therefore, even if an application is defined with the outbound data compression option set to No, this setting is ignored if the default and alternate screen sizes of the virtual session do not match the default and alternate screen sizes of the physical session.

- Outbound compression is automatically turned off when:

  - The datastream contents prohibit the use of outbound data compression.
  - Only one primary window is in use.
  - The screen characteristics of the virtual session match the screen characteristics of the physical device session.

  The following commonly used datastream elements cause compression to be turned off automatically:

  - vector graphic displays
  - explicit partition operation
  - file transfer operation

  If the datastream requires compression (for example, if there is more than one window on the display), CL/SUPERSESSION attempts to update the virtual session's screen image buffer (vector graphic information, for example, cannot be mapped into the screen image buffer) and signals the outbound data compression routines to update the physical display screen with the data successfully mapped into the screen image buffer.

  Automatic disabling of compression is intended to allow host applications to function normally in a virtual session environment without the user or administrator having to check or set options for the virtual session.

  If a user has trouble with a particular application when operating through a virtual session, make sure that compression can be turned off automatically when this application is running. Try running the application with only one window on the physical session, and verify that the default and alternate screen sizes of the virtual session match those of the physical session.

- Outbound data compression is set individually for each virtual session. You can set the option when you define the virtual session with an APPLDEF command or session profile, and you can dynamically change the setting by executing administrator functions from the Main Menu action bar while the virtual session is active. Candle also provides a default trigger (\o) that allows the user to change virtual terminal options.

  CL/SUPERSESSION also supports outbound data compression for virtual session datastreams that use 3270 extended field and character attributes. The compression support is limited to the color, highlighting, and symbol set attribute types.

For information on performance implications of compression, see “Performance Tips” on page 229.

Inbound Compression

Inbound data compression also uses the virtual device buffer that represents the screen image.

With inbound compression in effect, when the application sends a message to the terminal and has pre-modified some of the data fields by setting the modified data tag on, CL/SUPERSESSION holds the modified data tag in the screen image buffer but does not send the modified data tags out to the physical device. When the terminal user finally enters an input message, the physical device transmits *only* fields modified by user keystrokes. CL/SUPERSESSION receives this input, adds the application's pre-modified fields (if they were not part of the user's input), and forwards the composite message to the application. The composite message is identical to the input that would normally be received from the device if the modified data tags had been propagated to the physical device.

Applications that send large amounts of pre-modified data to physical terminals may benefit from using inbound compression.

Note: Inbound compression can be used with most applications. However, users who require support for an erase input key, light pen, or cursor select key must set inbound data compression to No.

For information on performance implications of compression, see “Performance Tips” on page 229.

Invalid Code Point Translation

CL/SUPERSESSION implements invalid code point translation. This function is similar to the 3*n*74 control unit Miscellaneous Feature Options provision for Unsupported Control Code Translate. Code points are translated only when sent as display characters. Transmission of the code points by a host application is generally the result of a programming error. When these same values are sent as part of an order, they are processed normally. For example, a X'01' could be valid as part of a 3270 Set Buffer Address order.

The following hexadecimal values are changed to X'60' (a hyphen or dash character) when received from a host application on a virtual session:

- 01,02,03,04,06,07,09,0A,0B,0E,0F
- 10,14,16,17,18,1A,1B,1F
- 20,21,22,23,24,25,26,27,2A,2B,2D,2E,2F
- 30,31,32,33,34,35,36,37,38,39,3A,3B,3D,3E,3F

Translation ensures that the virtual session's screen image can be sent to the physical device without causing program checks at the device. You can use the IX option of the VSSOPT function to inhibit the translation in datastreams sent to the physical device with outbound data compression turned off.

Chapter 8. Product Dialog Customization

This chapter introduces some commonly used dialog customization techniques. It is not intended as a programmer's guide or dialog language reference. For that kind of information, see the *SSPL Programming Guide* and the *Dialog Language Reference Manual*.

The chapter defines the basics of dialog structure, and uses examples to illustrate dialog customization. You will learn how to

- code continuation characters
- modify a display
- modify PF12
- disconnect a locked terminal
- automate application logon
- implement a dialog

Note: CL/SUPERSESSION uses a naming convention whereby prologue sections end in P, epilogue sections in E, and English-language display sections in 1 (to support national languages).

Customizing the dialogs supplied with CL/GATEWAY and CL/SUPERSESSION is optional. If these product dialogs require modifications, Candle recommends that you install your changes using SMP/E USERMODs. A sample USERMOD can be found in the SAMPLEs library. SMP/E automatically notifies you if maintenance performed by Candle affects any modules you have customized.

Password Encryption

All password variables are now encrypted. This prevents passwords from being browsed in virtual memory and provides some protection against administrators and operators viewing users' passwords during logon. If you intend to use dialogs from earlier versions of the CL products to write your own dialogs, or to customize dialogs, make sure that your dialogs use the ENCDEC dialog function to encrypt passwords because existing product dialogs require password encryption. For instructions, see the *Dialog Language Reference Manual*.

Introduction to Dialogs

Dialogs are on-screen panels and associated logic that simplify the interaction between end users and applications. Dialogs may prompt for input, provide helpful information, and validate input to keep erroneous data out of the application system.

CT/Engine contains a dialog manager that allows you to customize or develop dialogs between VTAM users and applications. The function of the dialog manager is twofold. It lets you present the application to your users, and it provides a programming structure that lets you customize the interface to increase ease of use of the application. Complex or unusual logon procedures, command entry formats, or data manipulation procedures can be programmed for the user.

Understanding a Dialog

Figure 14 shows the code for a typical dialog. This dialog starts the logon process and displays a message indicating that the logon is proceeding. Ignoring the data definitions and comments, a dialog consists of three basic sections:

Prologue The section that does the processing before display (lines 001 through 004).

Body The section that defines the display (lines 005 through 012).

Epilogue The section that does the processing after the display (line 013).

Other sections are used to copy one dialog into another, insert comments, and define data.

Note: The declaration, prologue, and epilogue sections in the following example are not shown. They are contained in dialogs KLSATTR1, KLSUINIP and KLSUINIE, which are included by using the)copy statement.

```
001 )copy KLSATTR1
002 )attr
003 '@' type(output) color(yellow)     display(high)   highlight(blink)
004 )copy KLSUINIP
005 )body top
006 +&panid
007
008 * Establishing default applications  *
009       environment for &vssuser
010
011 @       Please stand by.
012 <
013 )copy KLSUINIE            /* non-language dependent code  */
```

Figure 14. Typical Dialog

Line 001: Copies in the member KLSATTR1 (from the panel library, *&rhilev*.RLSPNLS), which contains all the CL/SUPERSESSION standard defaults for attributes. See “Changing the Color and Other Attributes” on page 129.

Line 002: Specifies the start of an attribute division. See “Changing the Color and Other Attributes” on page 129.

Line 003: Defines the at sign (@) as an attribute character to define a blinking output field, highlighted (if the terminal supports highlighting) and yellow (if the terminal supports color). See the *Dialog Language Reference Manual*.

Line 004: Copies in the member KLSUINIP, which contains the prologue of this dialog.

Line 005: Starts the body section, which defines the display.

Line 006: Uses the plus sign (+), specified in KLSATTR1 as an attribute character (blue, normal output; see Figure 18 on page 129), and displays the variable PANID, which contains the name of the panel displayed.

Line 007: Specifies a blank line.

Line 008: Displays the text `* Establishing default applications  *`.

Line 009: Displays the text `environment for`. The Dialog Manager resolves the variable VSSUSER, which contains the user ID.

Line 010: Specifies a blank line.

Line 011: Uses @ as an attribute character (defined in line 003), and displays the text `Please stand by`.

Line 012: Uses the less-than sign (<), specified in KLSATTR1 as an attribute character (turquoise, normal output; see Figure 18 on page 129).

Line 013: Copies in the member KLSUINIE, which contains the epilogue of this dialog.

Naming Variables

Dialog Language statements and operators are treated as reserved words. Do not use them as variable names. If you do, unpredictable results may occur. See the *Dialog Language Reference Manual* for the most current information on reserved words.

Continuation Characters

The plus (+) and minus (-) signs can be used to extend a command line or text string beyond the length of a single line. If you include a + character to the right of a line, all blanks to its left, as well as any leading blanks on the following line, are preserved. In the example below, the message text covers two lines. Since there is one blank to the left of the +, and two leading blanks on the next line, three blanks will be inserted in the text before the word RETURN.

```
set vssmsg '  SS904 UNABLE TO CREATE SESSION &sysparm, +
  RETURN CODE = &rc'
```

To join two lines with only a single blank space, use the - character. If you use a - character instead of a + in the above example, blanks to the left of the - character, as well as those preceding any text on the following line, are replaced with a single blank character. Because a blank is always inserted, the - character can be used to join words in a sentence, but not characters in a word.

Important

Be careful not to leave extraneous continuation characters at the end of a line. This can result in unexpected concatenations and interrupt the program.

If ISPF Number Mode is on, columns 73–80 contain line numbers. If you create a member in *&rhilev*.RLSPNLS, make sure that Number Mode is off, to match other *&rhilev*.RLSPNLS members. Otherwise dialogs will fail, since the Dialog Manager interprets all 80 columns as free-form data.

In *&rhilev*.RLSCMDS and *&rhilev*.RLSPARM, line numbers in columns 73–80 do not cause problems unless the member contains a mixture of both numbered and unnumbered lines. Therefore, make sure that all lines within a member are either all numbered or all unnumbered.

Modifying a Display

The simplest part of a dialog to customize is the text that is displayed. Suppose you want to customize the banner. This is the first line on the first display that the user sees. Typically, the banner includes a title identifying the purpose of the display, the department that is authorized to use the product, or steps that the user must take to proceed. The standard title used by CL/SUPERSESSION is Entry Validation (see Figure 15).

```
KLGLGON1 -------------      Entry Validation    ----------------------
```

Figure 15. Example Display Banner

Figure 17 shows the part of the KLGLGON1 dialog that defines the banner. The title is specified by the variable TITLE, which contains the text that appears on the screen (see Figure 16).

```
set cfpdata1 (REPEAT('\01',16))           /* build top display line */
set cfpdata2 (REPEAT('\01',22))           /* build top display line */
set panid (cstack())                      /* build top display line */
)prologue
```

Figure 16. Part of Dialog KLGLGONP that Defines the Title

```
)body top
+&panid <&cfpdata1        *&title                         <&cfpdata
 Date:&sysdate                                               System: &syssmfid
 Time:&systime                                               Device: &systerm
```

Figure 17. Part of Dialog KLGLGON1 that Defines the Banner

The variable TITLE is specified in the dialog KLGLGONP (see Figure 16 on page 128).

You change the title by editing the text between the single quotes. As soon as you refresh the dialog (see “Implementing a Dialog” on page 133), the new title will appear for any user.

Changing the Color and Other Attributes

Special characters, called *attributes*, control the color and intensity of the terminal display. Figure 18 lists the standard attributes used by CL/SUPERSESSION.

An attribute defines certain characteristics of the display. These include the presence of input and output fields, color, and highlighting. For example, a semicolon (;) specifies that an input field will follow, the input characters will appear in green (if the terminal supports color), in normal intensity, and will be underscored (if the terminal supports underscoring). For more information on attributes, see the *Dialog Language Reference Manual* and the IBM Manual *3270 Information Display System Data Stream Programmer's Reference*.

```
)ATTRS
';' type(input)  color(green)      display(normal) highlight(underscore)
'_' type(input)  color(yellow)     display(high)   highlight(underscore)
'%' type(input)  color(green)      display(invisible)
'*' type(output) color(yellow)     display(high)
'<' type(output) color(turquoise)  display(normal)
'>' type(output) color(turquoise)  display(high)
'+' type(output) color(blue)       display(normal)
'=' type(output) color(white)      display(high)
'?' type(input)  color(green)      display(normal)
'"' type(skip)   color(turquoise)  display(normal)
```

Figure 18. Sample Member Containing CL/SUPERSESSION Standard Attributes

Note: Changing this member modifies all of the screens displayed.

If you want to display characters used to define attributes (for example, the symbols in the second column of Figure 18), you must either eliminate them as attribute characters or replace them with characters you do not need to display.

For example, suppose you want to display the question mark (?), which is an attribute character, in a dialog. If the exclamation point (!), or some other character, is not being used to define an attribute, you can use it to replace the question mark. Make the substitution as follows:

1. Code an attribute section.
2. Copy the line containing the attribute character you want to change.
3. Replace the attribute character.

```
)attr
'!' type(input)  color(green)      display(normal)
```

This frees the question mark for use as a normal display character. After you use the question mark code the following to return the question mark to CL/SUPERSESSION standard usage as an attribute (see Figure 18 on page 129):

```
)attr
'?' type(input)  color(green)      display(normal)
```

Changing the Format of a Table

You can change the order of columns in a table display. Figure 19 is a sample display of a table of information, and Figure 20 shows the code that produces the display.

```
 Select triggers with a / or an action code.

    Phrase    Key     Dialog     Parameter
    -------   -----   --------   --------------------
    \q        ENTER   KLSQUIT
    \l        ENTER   KLSLOCK
    \m        ENTER
```

Figure 19. Example of a Table Display

```
<Select triggers with a*&vs<or an action code.

>   Phrase    Key     Dialog     Parameter
<   -------   -----   --------   --------------------
)body table input
;s= &vsptdph  &vsptdky&vsptddg   &vsptdpr
```

Figure 20. Code from Dialog KLSTRG11, Used to Display the Table

To have the dialog column follow the phrase column, move the column header (Dialog) and its variable (vsptddg) two fields to the left. Figure 21 shows the new display and Figure 22 the code that produces it.

```
 Select sessions with a / or an action code.

    Phrase    Dialog    Key      Parameter
    -------   --------- ------   --------------------
    \q        QUIT      ENTER
    \l        LOCK      ENTER
    \m                  ENTER
```

Figure 21. Table Display after Modification

```
<Select triggers with a*&vs<or an action code.

>   Phrase    Dialog    Key      Parameter
<   -------   --------- ------   --------------------
)body table input
;s= &vsptdph  &vsptddg&vsptdky   &vsptdpr
```

Figure 22. Code Modification in Dialog KLSTRG11

Modifying PF12

Depending on the number of PF keys you have on your keyboard, you may want to modify the PF10 key so it functions in the same way as the PF12 key. To do this, enter the following in the panel you want to change.

```
)prologue
set msgarea ''
set keystr (fold '&syskey = 'F10' set syskey 'PF12'
if &syskey = 'PF10' or &syskey = 'F10' set syskey 'PF12'
/* above line inserted */
if ((substr '&syskey',0,2) = 'PF')
```

Figure 23. Code Modification for PF12 Key

This will not change the display, but the PF10 key will act as the PF12 key.

Disconnecting a Locked Terminal

As part of profile administration, the administrator sets a timeout interval. This interval is the amount of time that a terminal can be idle before it is locked. The terminal remains locked and connected until the user enters a password to unlock it.

You can set a second timeout interval that logs off the locked terminal if the user takes no action after a defined period of time. (This does not affect users who are allowed to preserve active sessions upon exiting.)

To implement the second timeout interval, modify dialog KLSUNLKP as follows:

1. After the following code in KLSUNLKP:

   ```
   )prologue
      set unlock ' '
      set  syscsr unlock
      if (&systimeo) or (&vsptodlg)
         TIMEOUT(0)
   ```

   add the following new code for the second timeout interval:

   ```
   / *New code to add second timeout interval*/

      if not (&systimeo)   /* Is this an actual timeout? */
         TIMEOUT(0)        /* No */
      else
         TIMEOUT(&vsptoint:00)  /* Yes */

   /* End new code */
   ```

2. To activate this change, be sure to refresh dialog KLSUNLK*n* where *n* is the language designation (English = 1).

Automating Application Logon

Candle provides sample dialogs to help automate the logon procedure for several applications in *&thilev.*TLSPNLS.

Application	Dialog for Automatic Logon
TSO	KLSONTSO
VM	KLSONVM
CT/Engine operator facility (VTPOPER)	KLSONVTP
CICS	KLSONCIC
OMEGAMON for MVS	KLSONOM

To use one of these dialogs, specify it as the initial dialog in a profile. Alternatively, you can specify **`INITDLG=dialogname`** in the APPLDEF statement that defines the application to CL/GATEWAY where **`dialogname`** is one of the automatic logon dialogs. For more information about the APPLDEF command, see the *Operator's Guide*.

When a user selects the application, the user ID and password entered at logon will pass to the application without being retyped.

Control Dialog

The control dialog is identified to CL/SUPERSESSION by VSSENTRY, and to CL/GATEWAY by VIGENTRY. As the name implies, the control dialog controls the session for the user. The control dialog is given control of the session when any of the following conditions occur.

- If the highest-level panel in a dialog returns to the dialog manager, or if the EXIT statement is issued from within any panel in a dialog, control will be given to the control dialog if a CL/SUPERSESSION foreground session is not defined. If a CL/SUPERSESSION foreground session is defined, that session receives control.
- When a CL/SUPERSESSION foreground session terminates, the control dialog controls the session unless a termination dialog has been specified in the application definition or in the session profile. If a termination dialog has been specified, that dialog receives control.
- If any of the following key sequences are detected, the control dialog controls the session. (multisession applications only):
  - if the Attn key is pressed (SNA terminals only)
  - if the Sysreq key is pressed twice in succession (SNA terminals only)
  - if a trigger is entered, and the trigger is defined to CL/SUPERSESSION with either a blank or a null dialog name

The control dialog can execute any CL/SUPERSESSION or CL/ENGINE function statement. However, in a standard environment, the control dialog usually presents the user with a list of sessions that are available for their selection.

Customizing Variables

&rhilev.RLSPNLS(KLSSDCL) declares all CL/SUPERSESSION and CL/GATEWAY session variables, and comments in this member explain the variables.

Important

Do not modify the KLSSDCL dialog; use it for reference only. Product variable modification may corrupt the way in which the product operates, thereby producing possible undesirable results.

You can find further information about dialog variables in the *Dialog Language Reference Manual*.

Implementing a Dialog

If you have modified and saved a dialog in *&rhilev*.RLSPNLS, use the REFRESH command to compile it and bring it into memory. Enter REFRESH from the CT/Engine operator console or from an MVS console.

In the following example dialog DLOGXY is REFRESHed.

```
REFRESH PANEL DLOGXY
```

In the following example, the CT/Engine jobname is KLV. REFRESH, entered at an MVS terminal, refreshes dialog DLOGXY:

```
F KLV,REFRESH PANEL DLOGXY
```

Chapter 9. CT/Engine Customization

This chapter explains the following customization options for CT/Engine.

- defining the CT/Engine operator facility
- CT/Engine operator facility signon panel
- defining a network entry point with the DIALOG command
- dialog manager options in KLVINDM
- presentation space manager initialization parameters in KLVINPSM
- global sense table support in KLVINSNS
- defining the table database in KLVINTB
- VIEWLOG database allocation and initialization parameters in KLVINVLG
- VTAM program operator (VPO) initialization parameters in KLVINVPO
- virtual session manager initialization parameters in KLVINVSM
- VTAM options initialization parameters in KLVINVTM

Defining the CT/Engine Operator Facility

The CT/Engine operator APPL is the primary logical unit to which CT/Engine system operators must log on or be connected, if they are to use the CT/Engine operator facility. The CT/Engine operator facility is a full-screen application that provides an interface to issue CT/Engine, VTAM (by the VPO command), MVS/JES (by the MVS command), and application subsystem commands.

VTAM commands are accepted only when the CT/Engine program operator (VPO) facility is active. MVS commands are accepted only if the CT/Engine job step is APF-authorized. MTO commands are accepted only if CL/GATEWAY for IMS is installed and customized.

NAM can provide a special authorization process for CT/Engine operators, to limit access to authorized operators only, and to limit issuance of certain commands to subsets of the authorized operator group. For more information about NAM, see “Network Access Manager” on page 147.

To activate the CT/Engine operator facility, issue a NODE command identifying the ACB name of the CT/Engine operator APPL. Candle recommends that you place the NODE command in a CLIST, to be invoked automatically at startup. A sample NODE command is provided in *&rhilev*.RLSCMDS(KLSSTART).

Note: You can define more than one CT/Engine operator facility, by first defining an ACB in SYS1.VTAMLST(*newname*) and then issuing a NODE command for each operator facility.

Here is the format of the NODE command:

```
NODE acbname [TIMEOUT=hh:mm:ss|interval_in_seconds|0]
```

acbname	Specifies the ACB name of the CT/Engine operator APPL, as defined in SYS1.VTAMLST(*newname*).
TIMEOUT	Specifies the length of idle time before automatic logoff from the CT/Engine operator facility. If you omit TIMEOUT or specify it as zero, no timeout is enforced, and idle operator sessions are not terminated.

CT/Engine Operator Facility Signon Panel

When an authorized CT/Engine operator logs onto the CT/Engine operator APPL, a full-screen signon panel is presented for user ID and password authorization. The name of this special panel is always KLVENTRY, and it must reside in *&rhilev*.RLSPNLS.

A default panel definition is distributed with the CT/Engine product; however, you can customize this panel by using the SSPL language as defined in the *Dialog Language Reference Manual*.

Defining a Network Entry Point with the DIALOG Command

The DIALOG command defines a network entry point for a dialog-based application. Processing logic and presentation formats are established in panel libraries.

More than one DIALOG command can be issued from a single CT/Engine environment. Each DIALOG command must refer to a unique ACB name. You can issue DIALOG commands at any time from the CT/Engine operator facility.

To stop a DIALOG process, issue the CLOSE command for the ACB name of the DIALOG application. This terminates the physical and virtual sessions of all users who are logged onto that application. To restart the dialog, reissue the DIALOG command. Users will have to log on again.

If a panel referenced by the DIALOG command is modified during CT/Engine execution, you can use the REFRESH command to update the dialog.

For detailed information on the DIALOG command, see the *Operator's Guide*.

Dialog Manager Options in KLVINDM

Dialog manager options can be controlled by parameters in *&rhilev*.RLSPARM (KLVINDM).

PRELOAD

PRELOAD specifies dialogs that are to be compiled at CT/Engine initialization.

```
PRELOAD dialog1 dialog2...
```

dialog The name of a dialog to be compiled in order to be available after CT/Engine initialization when requested by a user. Any dialog not preloaded will be dynamically refreshed the first time it is requested by a user.

You can code multiple PRELOAD statements and multiple dialogs on each PRELOAD statement.

CT/Engine initialization will be delayed while the requested dialogs are compiled. If any of the dialogs cannot be compiled, CT/Engine initialization fails.

The default specifies that no dialogs are preloaded.

Presentation Space Manager Initialization Parameters in KLVINPSM

The ERPTEXT parameter provides a recovery facility for invalid input received by the physical terminal. Instead of terminating the session, CT/Engine displays the Error Recovery Pop-up (ERP) on the screen. Press Enter to terminate this pop-up and allow CT/Engine to recover.

ERPTEXT (in member KLVINPSM) defines the text for this pop-up. You can customize the text for languages other than English or to provide a suitable error explanation other than the default. The default is shown below:

```
TERMINAL INPUT ERROR.
PRESS ENTER TO RECOVER.
```

CT/Engine allows up to 24 lines of text. Text on each line must be enclosed in quotes and must not exceed 72 characters.

Note: ERPTEXT must be repeated for each line. For example:

```
ERPTEXT='   This is a sample of ERPTEXT.          '
ERPTEXT='       Press ENTER to recover.           '
```

Global Sense Table Support in KLVINSNS

Global sense table support provides an external mechanism for defining RPL sense code exception processing rules. The sense code rules determine how CT/Engine processes VTAM RPL sense information during a session. This facility allows you to manage nonstandard device and application processing characteristics.

Sense codes are data returned by a session partner in response to a request. They inform the sender of unacceptable requests to the receiver.

CT/Engine uses a static table of sense codes and associated actions. Global sense table support allows you to merge CT/Engine distribution sense rules with customized sense rules for your site. This facility is the single point of control for all sense code rule definition.

CT/Engine global sense table support provides:

- one internal table containing all sense code rules
- definition of sense code rules to control the exception processing of resources using either explicit or generic names
- loading of sense code rules stored in an external file
- ability to dynamically refresh the global sense table

Sense Rule Definition Statement

You can input site-specific sense rules using the REFRESH operator command or the *&rhilev*.RLSPARM member KLVINSNS.

REFRESH Command

To refresh the in-storage user global sense table located in *&rhilev*.SENSETBL(SNSIN) issue the following:

```
REFRESH S SNSIN &rhilev.SENSETBL
```

Refer to the *Operator's Guide* for more information on the REFRESH command.

KLVINSNS Member

You can code the following parameters in *&rhilev*.RLSPARM(KLVINSNS) to define sense rule statements.

Format

```
LUSTAT|EXRESP
SSENSE(xxxx)
(USENSE(xxxx))
FROMAPPL(name)|FROMLU(name)
action1, action2..., actionn
```

Parameters

LUSTAT	Indicates LUSTAT processing. LUSTAT may be abbreviated as L.
EXRESP	Indicates exception response processing. EXRESP may be abbreviated as E. The entry will be invalid unless either LUSTAT or EXRESP appears.
SSENSE	A 4-digit hexadecimal keyword parameter that defines the system sense code value.
USENSE	A 4-digit hexadecimal keyword parameter that defines the user sense code. USENSE defaults to zeros when no parameter is used.
FROMAPPL	A 1–8 character keyword parameter specifying the application LU that the sense block entry is built for. This parameter may include wildcard characters. There are no defaults for this parameter. FROMAPPL defines sense rules for sense codes received by a virtual terminal from an application, such as CICS.
FROMLU	A 1–8 character keyword parameter specifying the terminal LU that this sense block entry is being built for. May include wildcard characters. No defaults. FROMLU is used to define sense rules for sense codes that CT/Engine receives from a physical terminal. The name specified by the FROMLU keyword is the physical terminal name. ***Note:*** The FROMLU disposition applies to any LU in session with CT/Engine as an SLU.
ACTIONS	The keyword parameters that define action flag settings that describe the exception handling processing for a block entry. The following keywords are accepted:

CLEAR	Clears the session. CT/Engine issues the SNA CLEAR command followed by SDT if a CLEAR action is specified. See the IBM manual *SNA Reference Summary* for details.
DELAY	Causes a delayed READY TO RECEIVE state to be set. Valid when RESET is also requested.

	DELAY and IMMEDIATE keywords are mutually exclusive. If specified together, they will cause a syntax error.
IMMEDIATE	Causes an immediate READY TO RECEIVE state to be set. Valid when RESET is also requested. DELAY and IMMEDIATE keywords are mutually exclusive. If specified together, they will cause a syntax error.
LOGOFF	Causes unconditional termination of the session.
NODUMP	Requests that an RPL dump not be taken for this sense code. NODUMP does not suppress the RPL dump if KLSSYSIN parameter DEBUG(Y) is coded.
RESET	Causes the retransmission of the messages being rejected. You can specify an IMMEDIATE retransmission or a DELAY retransmission of 15 seconds.

Examples

The following are several examples of sense rule statements.

To define a sense rule for a sense code where the system sense is X'0821' and the user sense is X'0000', code the following. This rule defines a mask for application applids with CICS as the first 4 characters. When an LUSTAT with this sense code is received and the ACB name matches the mask, CT/Engine will perform the requested action.

```
LUSTAT,SSENSE(0821),FROMAPPL('CICS*'),CLEAR
```

To define a sense rule for a sense code where the system sense is X'0801' and the user sense if X'0000', code the following. This rule applies to all physical terminals with any character as the first character, followed by 327 and any characters to complete the name. When EXRESP is received from a terminal that matches this mask, CT/Engine will perform the requested actions.

```
EXRESP,SSENSE(0801),FROMLU('?327*'),RESET,DELAY
```

To provide for physical disconnection of DIALUP terminals when the DIALUP line is disconnected, create a member called KLVINSNS in *&rhilev*.RLSPARM and add the following statements:

```
LUSTAT,SSENSE(0802),FROMLU(TRM*),LOGOFF
LUSTAT,SSENSE(082B),FROMLU(TRM*),LOGOFF
LUSTAT,SSENSE(0813),FROMLU(TRM*),LOGOFF
```

where TRM* uniquely identifies the LU names of DIALUP terminals.

Create the following statements in member KLVINSNS of *&rhilev*.RLSPARM:

```
EXRESP,SSENSE(0813),FROMLU(?TRA*),CLEAR
EXRESP,SSENSE(081B),FROMLU(?TRA*),CLEAR
```

where ?TRA* uniquely identifies the LU names of the satellite terminals.

Usage Notes

1. Wildcard characters (* and ?) are supported.

   ***** When standalone, causes a match on any APPL or LU ID. When preceded by characters, matches any data for the length and portion of the string not explicitly defined.

   All characters following the * are ignored.

   ? When standalone, matches any one character name. When preceded or followed by characters, matches any character in the same relative position of the target string.

2. When syntax errors are encountered for sense rules defined in *&rhilev*.RLSPARM(KLVINSNS), CT/Engine initialization continues and the internal default table is loaded. Use the REFRESH command to correct the errors, and then update KLVINSNS.
3. The VSSTRACE operator command shows the mechanism the LU session partner is using (LUSTAT or EXRESP) in order to reject a message or to report status changes to the other half session.

Defining the Table Database in KLVINTB

The table database is a VSAM cluster used to store and retrieve permanent tables. These tables contain profile information such as session and triggers. Access is controlled by 128-byte keys.

Member KLVINTB of *&rhilev*.RLSPARM defines the table data base to the system. It contains the following parameters:

```
dsname -
disp [OLD|SHR] -
primarysize [nnn] -
secondarysize [nnn] -
minimumtubpool [nnn] -
maximumtubpool [nnn]
```

Note: If you specify keywords following *dsname* use the continuation character (-). KLVINTB reads only the first logical control statement.

Important

Do not modify the following parameters (with the exception of `disp`) except under the guidance of a Candle support representative.

dsname	Identifies the VSAM cluster name of the table database dataset.
disp	Specifies the disposition to be used when allocating the table database. ***Note:*** Specifying `SHR` will cause VSAM cluster corruption if the cluster is opened in write mode on more than one CPU.
primarysize	Initial amount of storage to be allocated when a table is created. Default is 2048. Minimum is 512. Maximum is 32767.
secondarysize	Amount of storage to be acquired when a table requires more storage than is available in its pool. Default is 2048. Minimum is 512. Maximum is 32767.
minimumtubpool	Minimum number of internal control blocks to be maintained in an available pool. Default is 20. Minimum is 12. Maximum is 248.
maximumtubpool	Maximum number of internal control blocks to be maintained in an available pool. Default is 100. Minimum is 20. Maximum is 256.

Example:

```
-rvhilev-.RLSTDB
```

To implement the table database facility, perform the following steps:

1. Allocate the table database cluster during CL/ENGINE installation.
2. Specify table database initialization parameters in member KLVINTB of *&rhilev*.RLSPARM.

For more information on the table database, see the *Dialog Language Reference Manual*.

Note: The table database cluster and all other CT/Engine VSAM files use local shared resources (LSR).

VIEWLOG Database Allocation and Initialization Parameters in KLVINVLG

The CT/Engine VIEWLOG database facility provides online access to selected categories of CT/Engine-generated messages, by way of an MVS console or the CT/Engine operator application. See the *Messages Manual* for more information on VIEWLOG console messages.

The log is maintained in the VSAM cluster you define. To implement the VIEWLOG database facility, perform the following steps:

1. Allocate the VIEWLOG cluster during CT/Engine installation.
2. Specify VIEWLOG initialization parameters in *&rhilev*.RLSPARM(KLVINVLG).

Member KLVINVLG of *&rhilev*.RLSPARM defines VIEWLOG to the system. It contains the following parameters:

```
dsname [RESET]
disp [OLD|SHR]
ddname [DDN]
```

dsname Identifies the VSAM cluster name of the VIEWLOG dataset.

RESET Specifies whether the VSAM cluster is defined as reusable. This parameter is optional.

disp The disposition to be used when allocating the viewlog file.

Note: Coding **SHR** will expose you to VSAM cluster corruption if the cluster is opened in write mode on more than one CPU.

ddname Controls how the file will be allocated.

If RESET is specified, the RBA is set to zero (0) during initialization so that the data space can be reused.

VIEWLOG event recording stops when an out-of-file-space condition is encountered. While the CT/Engine SYSOUT log dataset continues to record all messages, the log is not accessible through the CT/Engine operator facility.

Example:

```
-rvhilev-.RLSVLOG RESET
```

Usage Note

DSNAME and DDNAME control VIEWLOG database allocation.

Table 8. VIEWLOG Database Allocation

If you specify...	Then...
DSNAME only	The database is dynamically allocated.
DDNAME only	The cluster allocated to the specified DDNAME is used for the database, but there is no dynamic allocation.
DSNAME and DDNAME	The database is dynamically allocated using the specified DDNAME.
Neither DSNAME or DDNAME	VIEWLOG is disabled.

VTAM Program Operator (VPO) Initialization Parameters in KLVINVPO

If VTAM commands will be issued through the CT/Engine operator facility, a VTAM program operator APPL must be defined in SYS1.VTAMLST(*newname*). VPO initialization parameters must be defined to CT/Engine in member KLVINVPO of the initialization library (*&rhilev*.RLSPARM).

The parameters for VPO initialization are listed and discussed below:

```
acbname BUFLN(bufln) SHARE(opr) [PASSWORD(pswd)]
```

acbname	The ACB name of the VPO APPL as defined in SYS1.VTAMLST(*newname*).
BUFLN	The buffer length used between CT/Engine and VTAM. The recommended value is 160.
SHARE	The maximum number of outstanding VTAM operator commands. The recommended value is 16.
PASSWORD	An optional ACB password. This parameter is required only if an ACB password has been specified for the VPO ACB in SYS1.VTAMLST(*newname*).

This example uses the recommended values and continuation character (-) to initialize KLVLV002 184n.

```
KLVLV002   -
BUFL(160)  -
SHARE(16)
```

Virtual Session Manager Initialization Parameters in KLVINVSM

The KLVINVSM member allows certain CT/Engine virtual session manager (VSM) options to be specified. For more information on VSM, see the *Dialog Language Reference Manual*.

CLSDST-PASS Establishment Timeout

CT/Engine provides a default CLSDST-PASS virtual session establishment timeout of 60 seconds to prevent session hangs when a CLSDST-PASS fails for a virtual session.

To change the default, enter this statement as the first line of the KLVINVSM member:

```
TIMEOUT=time
```

Where *time* is either a numerical value that represents the number of seconds or a time value format hh:mm:ss to wait for the completion of a CLSDST-PASS operation.

Note: Virtual session establishment timeout can be disabled by specifying TIMEOUT=0.

For example, to set the CLSDST-PASS virtual session establishment timeout at two minutes, enter this statement as the first line of KLVINVSM:

```
TIMEOUT=120
```

VTAM Options Initialization Parameters in KLVINVTM

The KLVINVTM member allows certain CT/Engine VTAM options to be specified with the following parameters.

NOACQUIRESTEALING	Indicates that CT/Engine will not release an LU to a VTAM ACQUIRE command. When not specified, a VTAM acquire request will immediately be honored, and terminals will be unbound from existing sessions and released.
ACB31BIT	Indicates ACBs are located in extended storage. You must be running MVS/XA™ or MVS/ESA™, and be running MVS/DFP™ 3.2 or greater to specify ACB31BIT. Review IBM APAR OY13859.
ACB24BIT	Indicates ACBs are located in primary storage. This is the default.

RPL31BIT	Indicates RPLs are located in extended storage. This is the default.
RPL24BIT	Indicates RPLs are located in primary storage.
NIB31BIT	Indicates NIBs are located in extended storage. This is the default.
NIB24BIT	Indicates NIBs are located in primary storage.
SPT	Indicates session procedure timeout. CT/Engine provides a default session procedure timeout of 60 seconds, to prevent physical terminals from hanging up while waiting for sessions to start. To change the default, enter the following as the first statement of member KLVINVTM: `SPT=mm:ss`

The following example sets a session procedure timeout of two minutes and indicates that ACBs are located in extended storage.

```
SPT=2:00   -
ACB31BIT
```

Chapter 10. Network Access Manager

This chapter describes the functions of the Network Access Manager (NAM), and explains how to configure NAM and maintain the NAM database.

The Network Access Manager is one of the facilities supplied with CT/Engine. NAM provides:

- security system
- interface to an external security product
- database that stores user variables

The NAM records stored in a key-sequenced VSAM dataset make up the *NAM database*. When NAM is the chosen security system, the NAM database also holds encrypted security information. You can define a number of VSAM datasets to correspond to different NAM configurations, or you can use only one dataset for all configurations.

Even if you are not using the NAM database as your security system, we do not recommend deleting it. Some Candle products may store variables in the NAM database.

This section discusses the following topics:

- configuring the Network Access Manager
- maintaining the NAM database
- printing the NAM database
- user exit routines
- CA-ACF2 considerations

Configuring the Network Access Manager

To configure NAM, you must:

- define a VSAM dataset
- specify a security system
- declare variables to the VSAM dataset

Member KLVINNAM of TLVPARM contains an initial NAM configuration. This member defines a *control point*, that is, a set of parameters that designate a VSAM dataset name and a security system for access validation.

You can define more than one control point. All control points may use the same VSAM dataset to store security and profile information, or you can define a different dataset for each control point.

If you have not already done so, examine the control point currently defined in KLVINNAM, so that you know what your configuration looks like.

Defining NAM Control Points

NAM control points are defined during CT/Engine startup, by member KLVINNAM in TLVPARM. When a user logs on, the control point specified for the ACB takes effect and remains in effect until overridden by a CNTRLPT function in a dialog. For information on the CNTRLPT function, see the *Dialog Language Reference Manual,* LS53-3787.

Control points specify a security system, a VSAM dataset, and customized messages for the NAM exceptions that occur if you use the DB or RACF parameter.

The most commonly used security parameters and dataset name should be specified for the first control point definition in KLVINNAM, because this is the default control point. If a control point cannot be determined (that is, if no control point name matches the ACB name), CT/Engine uses the default control point, which is named DEFAULT.

Figure 24 shows the parameters that define a control point in member KLVINNAM of TLVPARM.

```
[control_point_name] [DSNAME=dsname]  -
                     [DDNAME=ddname]  -
                     [DISP=OLD|SHR]  -
                     [EXIT=exit]  -
                     [SAF|NOSAF]  -
                     [RACF|NORACF]  -
                     [DB|NODB]  -
                     [CLASSES=classes]  -
                     [NAF|NONAF]  -
                     [NOTIFY|NONOTIFY]  -
                     [LOG|NOLOG]  -
                     [APPL=appl]       -
                     [REQSTOR=reqstor]  -
                     [STAT|NOSTAT] -
                     [REUSEPW=8|reusepw]  -
                     [SUBSYS=subsys]  -
                     [MSG4='text']   -
                     [MSG8='text']   -
                     [MSG12='text']  -
                     [MSG16='text']  -
                     [MSG20='text']  -
                     [MSG24='text']  -
                     [MSG28='text']  -
                     [MSG32='text']  -
                     [MSG36='text']
```

Figure 24. NAM Control Point Parameters

control_point_name Specifies the control point name. To determine security for a particular ACB, use the VTAM ACB name specified in the HOSTGATE, NODE, or DIALOG command. If a matching control point name is not found for the ACB, NAM uses the DEFAULT control point (that is, the first control point defined in KLVINNAM).

If you define more than one control point, you must specify the control point name in all control point definitions except the first. If the control point name is omitted from the first control point definition in member

KLVINNAM in TLVPARM, the name DEFAULT is used.

DSNAME

Specifies the NAM database associated with the control point. If the DB parameter is specified, DSNAME or DDNAME must also be specified.

DDNAME

With DSNAME, controls how the database will be allocated:

> No DSNAME and no DDNAME. No database will be allocated.
>
> DSNAME and no DDNAME. The database will be dynamically allocated.
>
> DDNAME and no DSNAME. The cluster allocated to the specified DD name is used for the database; there is no dynamic allocation.
>
> DSNAME and DDNAME. The database is dynamically allocated using the specified DD name.

DISP

Controls the dataset disposition when the database is dynamically allocated:

OLD The database is allocated exclusively. No other jobs may allocate the database. If another job has the database allocated when CT/Engine starts, NAM reports that it cannot allocate the database and CT/Engine initialization terminates.

SHR The database is allocated shared. Other jobs may allocate the database.

Note: Coding DISP=SHR will cause VSAM cluster corruption if the cluster is opened in write mode on more than one CPU.

If DDNAME is specified with no DSNAME (in other words, no dynamic allocation is to be performed), the JCL dataset disposition will override any DISP value.

EXIT

Specifies an optional exit routine.

EXIT=exitname is called before RACF, SAF or DB, if specified. Validation may continue with other security methods depending upon the return code in R15 set by the exit. See “Control Point User Exit Routines” on page 161 for more information.

SAF|NOSAF

Specifies that the MVS System Authorization Facility (SAF) resolves NAM access and resource validation requests.

SAF overrides the RACF and DB parameters. The default is NOSAF.

	Note that when SAF is used, eight bytes of Common System Area (CSA) storage is required for each active, validated user. The location of this storage is dependent on the NAM Processing Option DATA=. For a detailed discussion of SAF, refer to IBM's *OS/VS2 SPL Supervisor* manual.
RACF|NORACF	Specifies whether to use RACF to resolve access requests. The default is RACF when RACF is present in the system; otherwise, the default is NORACF. RACF overrides the DB parameter and is overridden by the SAF parameter. If SAF is specified, NORACF is forced. Note that when RACF is used, eight bytes of Common System Area (CSA) storage is required for each active, validated user. The location of this storage is dependent on the NAM Processing Option DATA=.
DB|NODB	Specifies whether the NAM database associated with the control point will be queried for access validation. The default is DB if the DSNAME parameter is provided, otherwise the default is NODB. This parameter is overridden by both SAF and RACF. If DB is specified, DSNAME must also be specified.
CLASSES	Specifies an optional protected class list; that is, a member in TLVPARM that contains information used to construct the correct resource validation request for the security system. The NAM database (DB) does not use this feature. Refer to “Protected Class Lists” on page 154 for more information.
NAF|NONAF	Specifies whether to log a message in the Network Access Facility (NAF) dataset if any exceptions result from NAM access validation. The default is NAF.
NOTIFY|NONOTIFY	Specifies whether to log a message in the CT/Engine log dataset when an exception occurs during NAM access validation. The default is NOTIFY.
LOG|NOLOG	Specifies whether CA-TOP SECRET should or should not log an unsuccessful access to a resource.
APPL	(RACF and SAF Only) Overrides the APPL parameter used on the RACINIT or RACROUTE macro instruction. If not specified, the APPL parameter defaults to the ACBNAME of the current CT/Engine application. This parameter is ignored unless RACF or SAF is specified.

REQSTOR	(SAF Only) Functions exactly like the REQUESTOR parameter of the RACROUTE macro instruction. Refer to IBM's *RACROUTE Macro Rereference* manual for more information about the RACROUTE macro. This parameter is ignored if NOSAF is specified or implied.
STAT|NOSTAT	(SAF Only) Controls whether the statistics in the user profile should be maintained or not. STAT, the default, specifies that the statistics are controlled by the current SETROPTS options for the installation. If NOSTAT is specified, the statistics are not updated. In addition, no message is issued by RACF for a successful logon. Messages are always issued for unsuccessful logon attempts. This option can provide improved logon response times in a RACF environment by reducing IO to the RACF database and eliminating system enqueues. This is particularly beneficial with a shared RACF database. Refer to IBM's *RACF Security Administrators Guide* for more information about maintaining logon statistics. This parameter is ignored if NOSAF is specified or implied.
REUSEPW	(DB only) Specifies how often a user may reuse a previous password as a new password. Whenever a user changes his password, this number of prior passwords is checked. If the new password matches any of the previous ones, it is disallowed and the user must choose another new password. Specify a value from 0 through 8; 8 is the default. 0 means an existing password may be immediately reused as a new password.
SUBSYS	(SAF Only) Functions exactly like the SUBSYS parameter on the RACROUTE macro instruction. Refer to IBM's *OS/VS2 SPL Supervisor* manual for more information about the RACROUTE macro instruction. This parameter is ignored if NOSAF is specified or implied.
MSG4	Overrides the default message **KLVNA102 USER NOT DEFINED**.
MSG8	Overrides the default message **KLVNA103 PASSWORD NOT AUTHORIZED**.
MSG12	Overrides the default message **KLVNA104 CURRENT PASSWORD EXPIRED**.
MSG16	Overrides the default message **KLVNA105 NEW PASSWORD INVALID**.

MSG20	Overrides the default message **KLVNA106 USER NOT DEFINED TO GROUP**.
MSG24	Overrides the default message **KLVNA107 USER ACCESS REVOKED**.
MSG28	Overrides the default message **KLVNA108 GROUP ACCESS REVOKED**.
MSG32	Overrides the default message **KLVNA109 TERMINAL NOT AUTHORIZED**.
MSG36	Overrides the default message **KLVNA110 APPLICATION NOT AUTHORIZED**.

The EXIT, SAF, RACF, and DB control point parameters are resolved in the following order:

1. EXIT=exit
2. SAF
3. RACF
4. DB

If you use the EXIT parameter, the specified exit routine is invoked. Depending on the return code passed by the exit, access or resource validation may be complete, or control may be passed to SAF, RACF, or DB for further validation. If you do not use the EXIT parameter, the resolution order is as follows:

1. If SAF is specified with RACF, DB, or both, SAF controls access and resource validation for the control point.
2. If RACF is specified with DB, RACF controls access and resource validation for the control point.

Defining NAM Processing Options

Figure 25 shows the statements that may be used to define NAM processing options.

```
[DATA=BELOW|ABOVE]

[FIELDEXIT=fieldexit]
```

Figure 25. NAM Processing Options Statements

DATA Controls where the security control blocks are allocated by the CT/Engine NAM routines:

BELOW	Control blocks are allocated in 24-bit storage (RMODE 24).
ABOVE	Control blocks are allocated in 31-bit storage (RMODE 31).

When a control point exit routine is being used, bit #EPF1D31 in the #EPFLG1 flag byte of the $NAMUEPL parameter list may be examined to determine where security blocks should be allocated.

FIELDEXIT	Specifies a field-validation exit routine. Normally, CT/Engine requires that the data contain only printable characters. A user exit may be given control to validate each field using different requirements. See “Field Validation User Exit Routines” on page 165 for more information.

KLVINNAM Examples

Example 1: Using KLVINNAM to Specify More than One Control Point

In this example, the CT/Engine operator facility and GATEWAY1 are under the control of NAM; all other gateways are under the control of RACF. You might want to use this kind of setup to create two levels of security: one for data center personnel (under NAM), and one for all other users (under RACF, with dynamic application lists).

In addition, a field validation exit routine has been specified. This routine will get control to validate all data fields before they are passed to the control point exit and/or the MVS security system.

Define three control points in TLVPARM(KLVINNAM):

```
FIELDEXIT=KLVNAMPX
DEFAULT RACF CLASSES=external
KLVLV000 NORACF DB DSNAME=prefix.KLV.NAM
KLSGW001 NORACF DB DSNAME=prefix.KLV.NAM
```

Example 2: Using a Dialog to Specify More than One Control Point

In TLVPNENU, create a dialog containing this prologue:

```
)prologue
   CNTRLPT(cntrlptname)
      SET majorkey (VGET('&VSSUSER' var))
      VPUT('&VSSUSER' var 'str')
   CNTRLPT(DEFAULT)
```

where **cntrlptname** is the name of a control point defined in TLVPARM(KLVINNAM).

In this example, the first VGET and VPUT use **cntrlptname**, and subsequent VGETs and VPUTs use the DEFAULT control point. For more information on CNTRLPT, VGET, and VPUT, see the *Dialog Language Reference Manual*.

Example 3: Specifying More Than One NAM Database

You can set up more than one NAM database by specifying a different dataset name in each control point definition.

Define two control points in TLVPARM(KLVINNAM):

```
DEFAULT DSNAME(dsname1)
KLSGW002 DSNAME(dsname2)
```

where *dsname1* and *dsname2* are the names of two VSAM datasets to be used for two NAM databases.

If a user logs onto GATEWAY1, the gateway will use *dsname1* as the NAM database. If a user logs onto GATEWAY2 (ACB name KLSGW002), the gateway will use *dsname2* as the NAM database.

Protected Class Lists

You must define protected class lists if you intend to use either dynamic application lists or the RESOURCE dialog function. A *protected class list* is a member in TLVPARM that contains information used to construct the correct access request for the security system. This member is then specified on the CLASSES= keyword on a control point definition in member KLVINNAM.

A *dynamic application list* uses the security system to retrieve and build a list of authorized applications. The *Basic Configuration Guide*, LS55-3785, explains how to set up dynamic application lists through RACF, CA-ACF2, and CA-TOP SECRET. For information on the RESOURCE dialog function, see the *Dialog Language Reference Manual*.

The resource class names defined to the security system control access to specific resources (for example, physical terminal IDs). Protected class lists provide a connection between the resource classes defined to your security system and the resource classes defined to NAM.

Figure 26 shows the parameters of a protected class list.

```
class [EXTERNAL=external]  -
      [APPL=appl]  -
      [READAUTH=00|readauth]  -
      [REQSTOR=reqstor]  -
      [SUBSYS=subsys]
```

Figure 26. Protected Class List Parameters

class	Specifies a 1- to 8-character *internal* resource class name. This is the resource class name defined to NAM. The internal resource class name associated with dynamic application lists is VGWAPLST. You can define other internal resource names and invoke them with the RESOURCE dialog function.

EXTERNAL	Specifies a 1- to 8-character *external* resource class name. The external resource class name must match the resource class name defined to your external security system. (ACF2 only) ACF2 limits resource class names to four characters. The provided ACF2 exit (member KLVA2NEV of TLVSPENU) prefixes this name with the resource storage class character R. Therefore, your external resource class name for ACF2 can be only three characters long. (RACF only) The external resource class name must correspond to the CLASS parameter on the RACHECK (FRACHECK) macro.
APPL	(RACF/SAF only) Corresponds to the APPL parameter of the RACINIT macro.
READAUTH	(TOP SECRET only) Specifies the bit(s) in the TOP SECRET `INAACC` field that indicate read access. All bits must be on to indicate read access. If omitted or zeros, the `INAACC` field is ignored and the TOP SECRET return code is used to indicate read access. Must be two hexadecimal digits.
REQSTOR	(SAF only) Corresponds to the REQSTOR parameter of the RACROUTE macro.
SUBSYS	(SAF only) Corresponds to the SUBSYS parameter of the RACROUTE macro.

Maintaining the NAM Database

The NAM database can be maintained by either of two facilities:

- NAM command of the CT/Engine operator facility
- TLVPNENU dialogs

The four record types stored in the NAM database are described below. To map these records, use the $NAMR macro supplied in TLVMAC member KLV$$MAC.

Type of Record	*Description*
database control	This record contains information about the database, such as the date and time it was defined, and the date and time it was last accessed. Each database has one control record. The major key is 0, and the minor key is 0.
user control	These records are present only if the NAM database validates security for a control point (that is, if the DB parameter is specified in the NAM control point definition). User control records contain the password and expiration parameters. These records are created if the NAM SET command includes a password. The minor key is 0.

variable control	These records contain the variable names associated with a NAM database. Variable control records are defined by the NAM DECLARE command. You must DECLARE each variable before you can refer to it in a NAM SET command. The major key is 0. The minor key is the variable name.
user variable	These records contain the variable names and text associated with the major key. The major key is defined by the NAM SET command, and is usually a user ID or a terminal ID.

For complete information on the NAM command, see the *Operator's Guide*, LS99-3789.

Printing the NAM Database

You can print the NAM database at any time, even while CT/Engine is running. Use members KLVASM, KLVNAMPT, and KLV@NAM of TLVSPENU.

1. If you want to print the database while CT/Engine is running, specify DISP=SHR on the controlpoint specification in KLVINNAM.
2. To assemble the print program in member KLVNAMPT and link it to TLVLOAD, use the source code in member KLVASM.
3. To print the NAM database, run the job in member KLV@NAM in TLVSPENU.

Here is an example of output from the NAM database print job.

```
*********************** DATA BASE CONTROL RECORD ***********************

DATA BASE DEFINED ON 10/10/88 at 17:51:23 BY JOBNAME     ON SYSTEM SYSA
    LAST ACCESSED ON 11/12/88 at 14:12:26 BY JOBNAME     ON SYSTEM SYSA


****************** NAM DATA BASE RECORDS FOR APAR01 ********************

VSPSD00   :TSOB
VSPSD01   :MSA
VSPSD40   :D
VSPSD41   :D
```

RACF and SAF Considerations

This section covers some of the options that are available when you use RACF or SAF.

The System Authorization Facility (SAF) provides an installation with centralized control over system security processing through a system service called the MVS router. The MVS router provides a focal point for all products that provide resource management. The resource management components and subsystems call the MVS router as part of security decision-making functions in their processing, such as access control checking and authorization-related checking. These functions are called “control points.” SAF supports the use of common control points across products and across systems.

SAF is the preferred security interface for CT/Engine and can be used by installations that have CA-ACF2 or CA-TOP SECRET as well as with RACF, without the need to have any NAM exits installed. Refer to your security product documentation for information regarding the use of SAF.

Securing CT/Engine Applications with RACF

When the VALIDATE dialog function is issued to validate a user's security credentials, CT/Engine sets the current ACB name as the APPL paramter of a RACINIT or RACROUTE REQUEST=VERIFY macro. By creating resources in the IBM-supplied APPL class you can control which users can use the application as they enter the system. Use the UACC operand of an RDEFINE command to define the default universal access to a particular application, and issue PErmit commands to permit access to individual users or RACF connect groups.

Specifying APPL= for a control point definition in KLVINNAM can be used to reduce the total number of profiles that need to be defined.

Note: CT/Engine does not RACLIST APPL class profiles. For performance reasons, and to allow application resource profile changes to be immediately available, consider using SETROPTS GENLIST or RACLIST processing for the APPL class. For further information, see *IBM RACF: Security Administrators GUIDE*.

RACF Secured Signon Support

If you have RACF V2.1 or higher or RACF V1.9.2 with APAR OY65281, you can use the Secured Signon feature of RACF which allows access validation to be performed using a single-use password substitute called a *PassTicket*. A CT/Engine application can accept a RACF PassTicket created by a remote authentication service on behalf of a user without any changes to the CT/Engine application. RACF uses the value passed on the APPL parameter of a RACINIT or RACROUTE REQUEST=VERIFY to retrieve a profile defined to the **PTKTDATA** resource class. This profile contains a secret secured signon key that is used to derive the PassTicket. Note that the default APPL parameter is the ACBNAME of the current application, although this can be overridden by specifying an APPL parameter for the NAM control point definition in KLVINNAM.

Generating a PassTicket

When CT/Engine is used to logon to applications via a virtual terminal, a logon dialog can be used to automate the logon process. The userid and password used to access the CT/Engine dialog application can be replayed to the virtual session to achieve Single Signon. The userid and password must be kept in storage to accomplish this, although the password is usually masked with a proprietary algorithm. When access to the CT/Engine dialog application is made with a PassTicket, or for sites requiring a higher degree of security than is offered by the proprietary masking algorithm, CT/Engine can generate a RACF PassTicket for use in a virtual session logon dialog using the PASSTICKET dialog function.

A successful VALIDATE function must be performed before the PASSTICKET dialog function can be used. This is normally done at logon to the CT/Engine application. If RACF PassTickets are to be used for virtual session logons, and a regular password is used at initial logon, then the password should be oblitered from memory as soon as possible. To determine whether PassTicket support is enabled for the CT/Engine application a PASSTICKET dialog function should be performed after initial logon. A negative return code indicates that the support is not enabled, otherwise the password variable may be deleted and the storage that it occupied eradicated.

If you wish to use PassTickets for virtual session logons, and eliminate the in-storage copy of passwords, the following example SSPL determines whether PassTicket support is enabled for the current control point, eradicates the password variable contents and deletes the variable. Note that in order to entirely erase a variable's contents, it must be replaced with data that is the same length as it's original contents.

```
if  (PASSTICKET('&userid'
                '&sysappl'
                '&systerm'
                'PassTkt')) LT 0
  do
  set 'pswd' (REPEAT(' ',(LENGTH('&pswd'))))
  set 'pswd' ''
```

Once the password is eradicated, single signon can only be accomplished using a PassTicket. All logon dialogs must issue the PASSTICKET dialog function to return a PassTicket which can be entered into a virtual session application logon screen in place of the password. For example:

```
set rc (PASSTICKET('&userid'
                   '&sysparm'
                   '&systerm'
                   'PassTkt'
                   'PTMsg'))
if rc=0
  vsstype('&sysparm' '&PassTkt')
else do
  dialog errmsg '&PTMsg'
  return
  end
```

A PassTicket has only a short useful life, and may only be used once, and only for the destination application and userid for which it was generated.

PassTicket NAM Exit

The PASSTICKET dialog function invokes a NAM exit with a function code of 36. A sample exit, KLVRFPTX, is provided that validates the requesting user, and calls the RACF service to generate a PassTicket.

The exit should be assembled and linked with the JCL provided in member KLV@ASM of &thilev.TLVSPENU, specifying a load library that is defined to the CT/Engine JCL procedure concatenation of DDNAME TLVLOAD. The following NAM user exit parameters are passed for a PASSTICKET request:

Table 9. NAM User Exit Parameter List for PASSTICKET call.

Field	**Description**
#EPFC	36 (#EPFCPTK), indicating PASSTICKET generation request.
#EPTERM	Address of source terminal id vector.
#EPAPPL	Address of eight byte source applid.
#EPDUID	Address of destination userid vector.
#EPDEST	Address of destination applid vector.
#EPAUB	Address of $NAMAUB
#EPPTKT	Address of returned PassTicket vector.
#EPMSG	Address of a returned message vector or 0.
#EPAUB	Address of Active User Block ($NAMAUB).
$AUACEE	Address of RACF ACEE created by logon, if in use.
$AUUSER	User ID of requesting user.

To ensure that only those users authorized to generate a PassTicket for a particular userid and destination application are allowed to do so, the sample exit provides a method of validating a PASSTICKET request. The method assumes that a RACF general resource class called **PTKTVAL** is defined to the RACF Class Descriptor Table (CDT) with the following attributes:

```
PTKTVAL ICHERCDE CLASS=PTKTVAL,
           ID=NNN,
           MAXLNTH=39,
           FIRST=ANY,
           OTHER=ANY,
           POSIT=76,
           RACLIST=ALLOWED,
           OPER=NO
```

An ID value must be chosen that does not conflict with any existing class. A completion code 4 will result from the assembly of the CDT because of the non-standard CLASS and POSIT values. A POSIT value of 76 is chosen to associate this class with the IBM-supplied class **PTKTDATA** which is used to carry secured signon application information.

The RACF router table must be updated to include the PTKTVAL resource class with the following entry:

```
PTKTVAL ICHRFRTB CLASS=PTKTVAL,ACTION=RACF
```

For these changes to take effect you must IPL the system specifying CLPA to rebuild the PLPA.

To enable RACF secured signon and CT/Engine PassTicket validation you must logon to TSO with a RACF system special attribute userid and issue the following commands:

1. Activate the IBM-supplied PassTicket resource class **PTKTDATA** and the CT/Engine PassTicket validation resource class **PTKTVAL**.

   ```
   SETROPTS CLASSACT(PTKTDATA,PTKTVAL)
   ```

2. The PTKTVAL class should be defined to support generic profiles:

   ```
   SETROPTS GENERIC(PTKTVAL)
   ```

3. The PTKTDATA and PTKTVAL resource classes should be RACLISTed in global storage (a dataspace):

   ```
   SETROPTS RACLIST(PTKTDATA,PTKTVAL)
   ```

Now you can define profiles to the PTKTDATA class to identify each application that may accept a PassTicket. The purpose of the profile is to associate a secret key with a particular application running on a particular system. Profiles are defined with the RDEFINE RACF command:

```
RDEF PTKTDATA applid
     SSIGNON(KEYMASKED(application_key)|KEYENCRYPTED(application key))
             [UACC(NONE)]
```

Note: In order to use KEYENCRYPTED application keys, the system must have a cryptographic product active.

The profile name **applid** is application dependent. For CICS applications it is the APPLID parameter specified in the SIT. For IMS it is the IMSID specified by IMSCTRL in an IMSGEN. For TSO, a profile must be created with the prefix “TSO” concatenated with the SMFID of the target system. For example, TSO running on a system with SMFID “SYSB” must be defined to the PTKTDATA class as **TSOSYSB**. CT/Engine applications will use the ACBNAME specified in the DIALOG, HOSTGATE or NODE command, or may be overridden with the APPL parameter of a control point definition in KLVINNAM.

The sample NAM exit validates a users' authority to generate a PassTicket for an application by examining the UACC field of general resource profiles in the PTKTVAL class defined with the following format:

Dest_Appl.Source_Appl.Req_Uid.Dest_Uid

Generic profiles may be used.

If a matching profile is not found, PassTicket generation authorization fails. The PASSTICKET dialog function will result in a return code 4. If a matching profile *is* found, the UACC field is examined. If the UACC is **UPDATE**, then a PassTicket is generated for the destination application userid. If the UACC is **READ**, a PassTicket is only generated if the destination application userid is the same as the requesting userid. All other combinations of userid and UACC will fail PassTicket validation. A message is issued to TLVLOG describing the reason for the failure, and is availble to the issuing dialog in the optional message variable which can be supplied on the dialog function.

Profiles may be defined using the applid of the destination application, or the session ID used in a VSSDEF or VSSLOGON statement. The name that is used must be the same as the DEST_APPL name used in the PASSTICKET dialog function call. If the APPLID used in the PTKTVAL profile is different to the application profile name in the PTKTDATA class, APPLDATA can be supplied on the PTKTVAL profile definition to provide the PTKTDATA profile name. This may be particularly useful for TSO profiles, since it is unlikely that TSO is defined with a name that PTKTDATA profiles demand.

The following example allows all users to generate a PassTicket for their current userid for TSO defined as “TSOB” running on a system with an SMFID of “SYSB.”

```
     RDEF PTKTVAL TSOB.** UACC(READ) APPLDATA('TSOSYSB')
```

The secured signon profile for TSOB may be defined as:

```
   RDEF PTKTDATA TSOSYSB
        SSIGNON(KEYMASKED(ABCDEF01233456789))
        UACC(NONE)
```

Whenever PTKTVAL or PTKTDATA profiles are added or modified, the in-storage profiles must be updated with the following command:

```
   SETROPTS RACLIST(PTKTDATA,PTKTVAL) REFRESH
```

CAUTION:
Issuing this command for classes with a large number of profiles may impact system performance.

Control Point User Exit Routines

This section describes the conventions you must follow when writing NAM exits for access validation, CT/Engine operator validation, and CT/Engine command authorization. You may also refer to:

- macro $NAMUEPL, provided in the TLVMAC member KLV$$MAC
- sample exits KLVA2NEV and KLVNAMX, provided in TLVSPENU

Conventions

The exit routine must be linked into a dataset that is part of the TLVLOAD DD concatenation.

The first word in the exit must be a NOP instruction containing the length of the save/work area CT/Engine is to allocate. The address of this area is passed in field #EPWORK of the $NAMFEPL parameter list. The area is always in 24-bit storage (RMODE 24).

The save area defined by the exit must be **19 fullwords**, not 18. The extra word will be used for any $USREXIT calls.

To allow access to a resource, the exit should do nothing. To deny access, the resource name ($RVPAUTH in the $NAMVRPL area) should be cleared to nulls.

The exit is entered in AMODE 31 and may switch between AMODE 24 and AMODE 31 as needed. It must return to AMODE 31 before exiting.

The exit may reside in either 24- or 31-bit storage (RMODE 24 or RMODE 31). However, the security system in use may require that the module be RMODE 24.

Otherwise, standard MVS linkage conventions are to be followed.

Registers on Entry

R15	Contains the exit entry point address.
R14	Contains the return address.
R13	Contains the address of the caller's save area.
R1	Points to the $NAMUEPL parameter list. Refer to “$NAMUEPL Parameter List” for details.

Registers on Exit

R15	Return code. Refer to “Exit Return Codes” on page 164 for possible values.
R14	Not used.
R1	Not used.
R0	Not used.

Registers 2 through 13 must be restored to their entry values.

$NAMUEPL Parameter List

The $NAMUEPL macro, provided in the TLVMAC member KLV$$MAC, defines the parameter list that is passed to a control point user exit. All fields are 4 bytes long, except as noted.

Table 10 (Page 1 of 2). $NAMUEPL Parameter List		
Field	**Offset**	**Contents**
#EPWORK	0	Points to the save/work area provided for the exit.
#EPFC	4	Contains a value that describes the NAM function being performed: **0** (#EPFCENT) Entry validation **4** Not used. **8** Not used. **12** (#EPFCOPR) CT/Engine operator validation. **16** (#EPFCCMD) CT/Engine command validation. **20** (#EPFCRVL) Resource list validation. **24** (#EPFCLGF) User logoff. **28** (#EPFCINI) Control point initialization. **32** Not used. **36** (#EPFCPTK) PASSTICKET request.
#EPBUF	8	For #EPFCCMD, contains the address of the command and its operands.
#EPLIST		For #EPFCRVL, contains the address of the list of resources. This list is mapped by the $NAMRVPL macro, in TLVMAC member KLV$$MAC.
#EPBUFL	C	For #EPFCCMD, contains the length of the command and its operands.
#EPCOUNT		For #EPFCRVL, contains the number of entries in the $NAMRVPL area.
#EPUSER	10	For #EPFCENT, contains the address of the user ID vector.
#EPOPID		For #EPFCOPR and #EPFCCMD, contains the address of the operator ID vector.
Note: A vector consists of a one-byte length, followed by the data.		
#EPPSWD	14	For #EPFCENT, contains the address of the password vector.
#EPVAR		Not used.
#EPCMD		For #EPFCCMD, contains the address of the command name vector.
#EPCLASS		For #EPFCRVL, contains the address of the resource class name vector.
#EPDUID		For #EPFCPTK, contains the address of the destination userid vector for a PASSTICKET request.
#EPNPSWD	18	For #EPFCENT, contains the address of the new password vector.
#EPDEST		For #EPFCPTK, contains the address of the destination applid vector for the PASSTICKET request.
#EPGROUP	1C	For #EPFCENT, contains the address of the group name vector.

Table 10 (Page 2 of 2). $NAMUEPL Parameter List		
Field	**Offset**	**Contents**
#EPACCT	20	For #EPFCENT, contains the address of the accounting information vector.
#EPPROC	24	For #EPFCENT, contains the address of the 8-byte procedure name. It will be left-justified and blank padded.
#EPTERM	28	For #EPFCENT, #EPFCOPR, #EPFCCMD, and #EPFCPTK contains the address of the 8-byte terminal name. It will be left-justified and blank padded.
#EPAPPL	2C	For #EPFCENT, #EPFCOPR, #EPFCCMD, and #EPFCPTK contains the address of the 8-byte application name. It will be left-justified and blank padded.
#EPCNTPT	30	For #EPFCENT and #EPFCOPR, contains the address of the control point name vector.
#EPMSG	34	For #EPFCENT, #EPFCOPR, #EPFCCMD, and #EPFCPTK contains the address of a message vector where the exit can return a message. The first byte must be set to the text length, and the text must begin immediately after the length.
#EPAUB	38	For #EPFCRVL, #EPFCLGF, and #EPFCPTK contains the address of the active user block. This area is mapped by the $NAMAUB macro, in TLVMAC member KLV$$MAC.
#EPFLG1	3C	A one-byte options flag. **x'80'** (#EPF1D31) Control blocks are to be in 31-bit storage.

Exit Return Codes

Initialization (#EPFCINI) Return Codes

When a control point is initialized (during CT/Engine startup), the user exit is invoked with #EPFC containing #EPFCINI. The exit must return the desired active user block ($AUB) user extension length in R15. (Most exits will return R15=0.) If present, this area begins at label **$AUEXT** and will be available for subsequent exit invocations.

All Other Return Codes

The following return codes may be passed to NAM from an exit for all invocations except initialization. Additionally, the #EPMSG field in the $NAMUEPL area may be updated to point to a message. The message pointed to by #EPMSG overrides any default message text.

Note that return codes, with the exception of 0, are equivalent to RACINIT return code values plus 4. For instance, a RACINIT return code of 8 is the same as a CT/Engine return code 12. For more information on RACINIT, refer to *OS/VS2 SPL Supervisor*.

One of the following return codes may be passed in R15 from the exit:

0	Continue security processing. If no other access mechanism is specified, the user is considered authorized. For #EPFCPTK the exit does not support PASSTICKET requests.
4	The user, operator, command, or PASSTICKET request is authorized.
8	The request could not be satisfied because of an invalid user ID or because of lack of operator, command, or PASSTICKET generation authorization.
12	For #EPFCENT, the request could not be satisfied because of an invalid password. For #EPFCPTK, a RACF PASSTICKET generation service request error occurred.
16	For #EPFCENT, the request could not be satisfied because of an expired password.
20	For #EPFCENT, the request could not be satisfied because of an invalid new password.
24	For #EPFCENT, the request could not be satisfied because the user is not defined to the group.
32	For #EPFCENT, the request could not be satisfied because the user's access has been revoked.
40	For #EPFCENT, the request could not be satisfied because the user's access to the specified group has been revoked.
52	For #EPFCENT, the request could not be satisfied because the user is not authorized to use this terminal.
56	For #EPFCENT, the request could not be satisfied because the user is not authorized to use this application.

Any return code other than the values specified will cause access validation, operator validation, or command validation to fail.

Field Validation User Exit Routines

This section describes the conventions you must follow when writing a NAM exit for field validation. You may also refer to:

- macro $NAMFEPL, provided in the TLVMAC member KLV$$MAC
- sample exit KLVNAMPX, provided in TLVSPENU

Conventions

The exit routine must be linked into a dataset that is part of the TLVLOAD DD concatenation.

The first word in the exit must be a NOP instruction containing the length of the save/work area CT/Engine is to allocate. The address of this area is passed in field #FPWORK of the $NAMFEPL parameter list. The area is always in 24-bit storage (RMODE 24).

The save area defined by the exit must be **19 fullwords**, not 18. The extra word will be used for any $USREXIT calls.

The exit is entered in AMODE 31 and may switch between AMODE 24 and AMODE 31 as needed. It must return to AMODE 31 before exiting.

The exit may reside in either 24- or 31-bit storage (RMODE 24 or RMODE 31).

Otherwise, standard MVS linkage conventions are to be followed.

Registers on Entry

R15	Contains the exit entry point address.
R14	Contains the return address.
R13	Contains the address of the caller's save area.
R1	Points to the $NAMFEPL parameter list. Refer to “$NAMFEPL Parameter List” for details.

Registers on Exit

R15	Return code. Refer to “Exit Return Codes” on page 167 for possible values.
R14	Not used.
R1	Not used.
R0	Not used.

Registers 2 through 13 must be restored to their entry values.

$NAMFEPL Parameter List

The $NAMFEPL macro, provided in the TLVMAC member KLV$$MAC, defines the parameter list that is passed to a field validation user exit. All fields are 4 bytes long.

Table 11. $NAMFEPL Parameter List		
Field	**Offset**	**Contents**
#FPFC	0	Contains a value that describes the NAM function being performed: **0** (#FPFCENT) Entry validation **4** (#FPFCVGT) VGET variable service. **8** (#FPFCVPT) VPUT variable service. **12** (#FPFCOPR) CT/Engine operator validation. **16** (#FPFCCMD) CT/Engine command validation. **20** (#FPFCRVL) Resource list validation. **24** (#FPFCLGF) User logoff.
#FPPTYP	4	Contains a value that describes the type of parameter being validated: **0** (#FPPTUID) User id. **1** (#FPPTPW) Password. **2** (#FPPTNPW) New password. **3** (#FPPTGRP) Group name. **4** (#FPPTACT) Accounting information. **5** (#FPPTPRC) Procedure name. **6** (#FPPTTRM) Terminal id. **7** (#FPPTAPL) Application id. **8** (#FPPTCPT) Control point name. **9** (#FPPTVAR) Variable name. **10** (#FPPTCMD) Command text. **11** (#FPPTCLS) Class name.
#FPWORK	8	Points to the save/work area provided for the exit.
#FPRSV1	C	Reserved for future use.
#FPPTADR	10	Points to the text to be validated.
#FPPTLEN	14	Contains the length of the text to be validated.

Exit Return Codes

One of the following return codes must be passed in R15 from the field validation exit.

0 The exit has accepted the parameter. CT/Engine will use the parameter as-is.

>0 The exit has rejected the parameter. The NAM function will be terminated.

<0 The exit has ignored the parameter. CT/Engine will continue with the standard validation.

CA-ACF2 Considerations

If you install Release 5.0 or later of CA-ACF2, you can take advantage of the following performance improvements:

- Extended User Authentication (EUA)
- Logon Inheritance (LI)
- Rule List Interpret (RLI)

The source modules for these features are in members KLVA2NEV and KLVA2INH of TLVSPENU.

Note: If you use the RESOURCE dialog function with ACF2, be sure to issue CNTRLPT and VALIDATE before RESOURCE. The CNTRLPT, VALIDATE, and RESOURCE function calls must all be in the same dialog. See the *Dialog Language Reference Manual*.

Extended User Authentication Support

This feature allows communication with other vendor security products through ACF2 in very secure environments. Through the CT/Engine ACF2 exit, a conversation takes place between ACF2 and the other security product until access is allowed or denied. In this way, all users who access the network through a CT/Engine-based product can be assured complete verification.

EUA uses the $USREXIT DIALOG service and TLVPNENU member KLVACE to display prompts and collect replies from the end user.

Installation

Member KLVA2NEV of TLVSPENU includes support for EUA. The ACFOPTS macro specifies **EUA=YES**, so users with the appropriate flags generated in their ACF2 logonid record will cause the EUA dialog to be invoked. For information on implemeting EUA support in ACF2 refer to the *CA-ACF2 System Programmers Reference.*

Implementation

Member KLVACE of TLVPNENU displays and collects information. Three variables are associated with the panel:

ACEUARPY	The reply variable. Information filled in by the user is passed to the other security product.
ACEUAPRM	The prompt variable. When the other security product requests information (through the ACVXBCNT field in ACVXAUB), the message returned is displayed in this variable.

ACEUA000-xxx	Information variables. Some security products return more than one message for display before requesting a reply. The messages are stored in this variable array. The panel is displayed when a reply is requested.

You may need to customize EUA. The following psuedocode explains how EUA operates:

```
initialize EUA environment
validate:
issue ACFSVC for entry validation
if (ACFSVC successful)
   if (EUA in progress)
      release any reply buffers allocated
      find message (from other vendor product)
      if (message exists)
         if (reply requested)
            save in ACEUAPRM
         else
            save in ACEUAxxx
            bump ACEUAxxx index
         endif
      endif
      if (dialog in progress)
         if (reply requested)
            do while (more replies)
               display KLVACE
               get reply (returned in ACEUARPY)
               chain reply buffer (for other vendor product)
               decrement reply count
            enddo
            goto validate
         endif
      else
         check whether user is authorized
      endif
   endif
endif
```

Logon Inheritance Support

This feature lets you replace the in-storage representation of a password with an *inheritance indicator* created from three elements:

- user ID
- source (terminal ID)
- resource (application)

Use of the inheritance indicator eliminates integrity exposures.

Note: Logon inheritance works only for local VTAM applications, not cross-domain. This is a CA-ACF2 restriction.

Logon inheritance is invoked by the LINK dialog language statement, most likely from a logon dialog. The TLVLOAD and STEPLIB libraries must be APF-authorized.

Installation

The source module for logon inheritance is in member KLVA2INH of TLVSPENU. Assemble the routine into the TLVLOAD target library, using the current level of TLVMAC, SYS1.ACFMAC, and SYS1.MACLIB. A sample jobstream is provided in member KLVASM. The SYS1.ACFMAC library needs to be concatenated in SYSLIB.

Implementation

The LINK dialog language statement invokes KLVA2INH to obtain the inheritance indicator, which is good for only one logon. Each time an application that has an auto-logon script is selected, the LINK statement can be used to create another inheritance indicator. You can then use this indicator, instead of the password variable, in the auto-logon sequence. Code the LINK statement as follows:

```
SET RC (LINK(KLVA2INH userid sourcename resourcename))
```

userid	The user ID for which the inheritance indicator is obtained.
sourcename	The source where the logon occurs. Normally, this is the virtual terminal network name.
resourcename	The name of the resource to which the source connects. This is the network name of the application being accessed.

The outcome of the function appears as a return code in variable &RC, which may contain one of these values:

0	Successful completion. Variable *&KLVA2INH* contains the returned token.
4	ACF2 not active.
8	Parameter error. One of the required parameters was not supplied.
12	ACF2 failed the request. Variable *&KLVA2INH* contains any error message returned by ACF2.

You may hard-code values for *userid*, *sourcename*, and *resourcename*, or you may use variables. To derive the variables:

userid	Use the variable *&VSSUSER*

sourcename Obtain the variable *&SRCNM* by placing the following statement before the LINK statement in your logon dialog:

```
SET SRCNM (VSSNODE(&SYSPARM))
```

resourcename Obtain the variable *&RSRCNM* by placing the following code before the LINK statement in your logon dialog:

```
SET RC (VSSVINFO(&VSPID))
IF (&RC > 0)
   DO
     VSSTERM(&VSPID 1)
     RETURN &RC
   END
SET RSRCNM '&VSSAPPL'
```

The LINK statement would now look like this:

```
   SET RC (LINK(KLVA2INH '&VSSUSER' &SRCNM &RSRCNM))
```

Rule List Interpret Support

Before RLI, a separate ACF2 generalized resource rule call (which generated an SVC) was required to validate access to each resource. Numerous calls were necessary to validate and construct a dynamic application list. RLI requires only one call to validate the entire list.

Installation

If ACF2 release 5.0 or higher is installed, RLI is automatically assembled into member KLVA2NEV of TLVSPENU. Global character symbols set by the invocation of macro @ID (in SYS1.ACFMAC) determine the release number of ACF2. During assembly, this information generates the correct object code in KLVA2NEV.

Implementing Group Profiles with ACF2

Note: This section applies only to CL/SUPERSESSION.

If ACF2 is performing your entry validation and you want to implement group profiles, you will need code a default group and assign it to users.

1. Have your security administrator define an 8-byte group field in the ACF2 Logon ID (LID) record. For example, the following statement defines the field GWGROUP in the user portion of the LID record.

```
@CFDE GWGROUP,LIDGWGRP,CHAR,ALTER=SECURITY+LEADER,GROUP=10,
      LIST=ALL,ZERO=YES,FLAGS=NULL
```

2. Add the following code to TLVSPENU(KLVA2NEV), the ACF2 user exit. During logon this code queries the value of the group field and sets the session variable MPCDFLT.

```
         CLI   LIDGWGRP,C' '
         BE    NOGRP
         $USREXIT PUTVAR,NAME='MPCDFLT',AREA=LIDGWGRP,LENGTH=L'LIDGWGRP
   NOGRP $
```

3. Insert the following code into the prologue of TLVPNENU(SUDEF), at the point where the group profile is resolved. During logon SUDEF sets up the user's profile variables.

```
   /* Open group profile.  If it doesn't exist, create one from NAM. */
   set vspdflt &vupdflt              /* Fold to upper case. */
   if not &vspdflt                   /* Got a value from user profile. */
     do
       if not &mpcdflt               /* If there is no value */
         set mpcdflt 'SSUSER'        /* set to SSUSER. */
       set mpcblank '&index('&mpcdflt',' ')'    /*Strip trailing blanks. */
       if &mpcblank lt 0
          set mpcblank 8
       set vspdflt '&substr('&mpcdflt',0,&mpcblank)'  /* Set up vspdflt. */
   end
```

   If you want the GLOBAL profile to be used as the default group profile, eliminate the assignment of SSUSER to *&MPCDFLT*.

If you are using an external security system other than ACF2, you can adapt this general procedure to fit your security system.

For more information on group profiles, see the *Basic Configuration Guide*.

Validating Only the ACF2 User ID

You can customize the KLSA2NEV exit to interrogate whether an ACF2 user ID is valid without checking the user's password. You can use this method to clean up unused user profiles in the TDB. To implement validating only the user ID, do the following:

1. Modify the KLVINNAM member of the initialization parameter library *&rhilev*.RLSPARM to include an additional ACF2 control point.
2. Create a new user exit by copying and modifying KLSA2NEV. You can refer to the sample modified exit member KLSA2NE1 in the *&thilev*.TLSSAMP library.
3. Assemble and link edit the newly created exit into the *&rhilev*.RLSLOAD target library.
4. Create a special dialog in the *&rhilev*.RLSPNLS dataset that checks only the user ID.

For more information, refer to the comments provided in sample exit member KLSA2NE1 in the *&thilev*.TLSSAMP library.

Important

The security exit is ***not*** driven during split window initialization.

Chapter 11. Network Accounting Facility

The Network Accounting Facility (NAF) keeps accounting and audit records at these levels:

- user
- terminal
- application
- session

The records may be directed either to a BSAM journal dataset or to the active System Management Facility (SMF) dataset. CT/Engine must be APF-authorized for SMF recording.

You can use the NAF data to

- monitor application and network resource usage
- evaluate system security measures
- perform network planning and capacity management
- provide input for network billing

CT/Engine, CL/SUPERSESSION, and CL/GATEWAY all generate NAF records.

NAF Startup Parameters

During CT/Engine initialization, the KLVINNAF member of the initialization library (*&rhilev*.RLSPARM) provides NAF startup parameters.

The format of the KLVINNAF parameters is shown below. All parameters must be provided as a single logical statement.

```
[DSNAME=dsname]
[BLKSIZE=nnnnn|3120]
[BUFNO=n|4]
[MOD]
[SMF=nnn]
```

DSNAME Identifies a sequential BSAM dataset in which NAF records are to be kept. The dataset must have the DCB attributes shown below. All other file characteristics are supplied internally.

```
DCB=(DSORG=PS,RECFM=U)
```

If DSNAME is omitted and SMF is specified, journal records are written directly to the active SMF dataset. If both DSNAME and SMF are omitted, no records are written.

BLKSIZE Specifies the block size for records in a dedicated journal dataset. This operand is optional. Values between 287 and 32767 are valid. The default is 3120.

BUFNO Specifies the number of buffers reserved for the BSAM access method with a dedicated journal dataset. This operand is optional. Values between 1 and 99 are valid. The default is 4.

MOD Specifies that NAF recording should begin after the last record already written to the NAF dataset. If you use the BSAM access method and omit MOD, recording begins at the beginning of the dataset and overwrites the existing contents. MOD is valid only for BSAM; it is not valid for SMF recording.

SMF Specifies SMF record type. CT/Engine must be APF-authorized for SMF recording. Values between 128 and 255 are valid. If SMF is not specified, no SMF records are written.

$NAFR Macro

You can use the $NAFR macro in member KLV$$MAC in the TGMACLIB library to generate a DSECT that defines the format of the NAF records. By using the date and time stamps provided in each record header, you can sort the records by date, time, and user ID to correlate all records for a specific user's session. For detailed information on NAF record formats, see “NAF Record Layouts” on page 179.

```
label $NAFR type,                                          X
            [DSECT=YES|NO],                                X
            [PREFIX=YES|NO]
```

label Specifies a three-character label to which standard record field names are appended. This operand allows many expansions of $NAFR in a single assembly. The default label prefix is NAF.

type Identifies the NAF record type. See “NAF Record Types” on page 177.

DSECT Specifies whether or not a DSECT statement should be generated. If not, a DS statement forcing double word alignment is generated. In either case, the label field consists of **cccDSECT** (where **ccc** is a three-character label). When defaults are used, a DSECT named NAFDSECT is generated. The default is DSECT=YES.

PREFIX Specifies whether or not the CT/Engine record header section will be included. NAF records generally include this prefix area. The default value, PREFIX=NO, suppresses the prefix.

NAF Record Types

This section describes the NAF record types for CT/Engine, CL/SUPERSESSION, and CL/GATEWAY. The keyword used to identify each record type is specified in the *type* operand of the $NAFR macro, to generate an assembler language mapping of the record.

CT/Engine Record Types

STARTUP	Subtype 00: System startup Generated at CT/Engine startup to account for the period of time that CT/Engine is active in the system.
SHUTDOWN	Subtype 01: System shutdown Generated at CT/Engine shutdown to account for the period of time that CT/Engine is active in the system.
EVR	Subtype 02: Entry validation Generated upon return from NAM, when invoked to validate the network access attempted by the user. If NAM is not active in the system, no recording is performed. The information includes the NAM access recommendation.

CL/SUPERSESSION Record Types

VSSULGN	Subtype 224: CL/SUPERSESSION entry Generated when a user establishes a session with CL/SUPERSESSION, after VSSENTRY is executed. The CL/SUPERSESSION entry and exit records contain the user ID, physical terminal ID, and the logical unit name of the entry point DIALOG APPL.
VSSULGF	Subtype 225: CL/SUPERSESSION exit Generated when a user logs off or is disconnected, or when the control dialog terminates.
VSSTLGN	Subtype 226: Virtual session initiation Generated when a new virtual session is established. The virtual session initiation and termination records contain the user ID, the application session logical unit name, the virtual terminal logical unit name, and a unique session resource identification.

VSSTLGF	Subtype 227: Virtual session termination Generated when a virtual session is terminated. This record contains the information mentioned in the virtual session initiation record.

CL/GATEWAY Record Types

GWLOGON	Subtype 240: CL/GATEWAY logon Generated when a user establishes a session with a gateway. This record is not dependent upon the acceptance of the user by NAM or another security system.
GWLOGOFF	Subtype 241: CL/GATEWAY logoff Generated when the user logs off the gateway.
GWPTSTRT	Subtype 244: CL/GATEWAY virtual session started Indicates that a SINGLE session was started between the user and the selected destination application. The information includes the destination application name and the logical unit name of the virtual terminal device selected.
GWPTSTOP	Subtype 245: CL/GATEWAY virtual session ended Indicates that a SINGLE session between the user and the selected destination application was terminated. The information includes the destination application name and the logical unit name of the virtual terminal device released.

NAF Reporting

&rhilev.RLSSAMP contains JCL and source programs that you can use to provide a formatted listing of the NAF journal dataset. If SMF recording is used, you can run the report against an unloaded SMF file, including one created by IFASMFDP.

The KLS@NAF member of *&rhilev*.RLSSAMP contains a sample JCL procedure and describes steps taken to install and execute the KLS@NAF report program. You can use the parameters described below to control the scope and format of the report. Enter the desired parameters in the SYSIN file of the NAF report program.

Note: If you use SMF recording, you must change the SYSINE15 DD statement to indicate that SMF input will be used.

DATE(begdate,enddate)	Specifies the beginning and ending dates of the range of records to process. Specify dates in dd/mm/yy format. If the DATE parameter is omitted, all NAF record dates encountered in the input file are processed.

TIME(begtime[,endtime])	Specifies the beginning and ending times of the range records to process. Specify times in hh:mm:ss format. If the TIME parameter is omitted, all NAF record times encountered in the input file are processed.
TYPE(type1,...,typen)	Specifies the record type codes of the records desired. If the TYPE parameter is omitted, all NAF record types are processed.
COUNT(nn)	Specifies the number of lines per page to be generated in the report. If the COUNT parameter is omitted, the default value of 50 lines per page is used.

Example: NAF Report Parameters

This example assumes that you are interested in determining the number of users handled by CT/Engine in a five-day period. The parameters shown below select all Entry Validation (EVR) records generated between March 21, 1993 and March 25, 1993. The report is printed using the default value of 50 lines per page.

```
DATE(03/21/93,03/25/93)
TYPE(EVR)
```

NAF Record Layouts

The $NAFR macro, which generates the $NAFR DSECT, provides assembler language mapping of each record layout. You must refer to this DSECT whenever you use the NAF reporting facility or write a custom NAF analysis/reporting program to obtain field labels and detailed record structures.

NAF records are grouped into blocks. The NAF block is the basic unit of transfer between the CT/Engine NAF recorder and the BSAM or SMF journal dataset. Each record block contains an 18-byte SMF-compatible header section, as described in the IBM manual *OS/VS2 MVS SPL System Management Facility*. The record type defined by the KLVINNAF SMF parameter resides in this section. The SMF header section is also present in NAF blocks when you use the BSAM access method.

The SMF header section is followed by one or more NAF records. Each NAF record consists of a fixed- length *prefix section* and a *record-dependent section*.

Member KLSNAF15 of *&rhilev.*RLSSAMP contains the source code for a routine that reconstructs individual records contained in a NAF block. The sample source routine is a sort input exit routine that passes deblocked records to the SORT utility. Each record produced by the sort contains the 18-byte SMF header, followed by the NAF record.

Member KLSNAFPT of *&rhilev.*RLSSAMP contains a basic NAF report program that illustrates the layout of a NAF record block and the use of the $NAFR mapping macro. The output from the sort utility is used as input to the print routine.

The following table shows the format of the NAF prefix, which is included in each record.

Field Name	Offsets	Length	Format	Description

RTYPE	0	1	binary	Record subtype:
				Hex: 00 = Startup record (10 bytes)
				01 = Shutdown record (10 bytes)
				02 = Entry validation record (219 bytes)
				E0 = CL/SUPERSESSION user logon (34 bytes)
				E1 = CL/SUPERSESSION user logoff (34 bytes)
				E2 = CL/SUPERSESSION terminal logon (38 bytes)
				E3 = CL/SUPERSESSION terminal logoff (72 bytes)
				F0 = CL/GATEWAY logon (34 bytes)
				F1 = CL/GATEWAY logoff (34 bytes)
				F4 = CL/GATEWAY SINGLE start (50 bytes)
				F5 = CL/GATEWAY SINGLE stop (50 bytes)

RTIME	1-4	4	binary	Time stamp (TOD clock)
RDATE	5-6	2	binary	Date stamp: Julian year
RDATE	7-8	2	packed decimal	Date stamp: Julian day
RLEN	9	1	binary	Variable data length

STARTUP Record

This record is generated at CT/Engine startup to account for the period of time that CT/Engine is active in the system. Its length is 10 bytes.

Field Name	Offsets	Length	Format	Description
RTYPE	0	1	binary	Hex: 00=Startup record subtype (10 bytes)
RTIME	1-4	4	binary	Time stamp (TOD clock)
RDATE	5-6	2	binary	Date stamp: Julian year
RDATE	7-8	2	packed decimal	Date stamp: Julian day
RLEN	9	1	binary	Hex: 00=Variable data length

SHUTDOWN Record

This record is generated at CT/Engine shutdown to account for the period of time that CT/Engine is active in the system. Its length is 10 bytes.

Field Name	Offsets	Length	Format	Description
RTYPE	0	1	binary	Hex: 01=Shutdown record subtype
RTIME	1-4	4	binary	Time stamp (TOD clock)
RDATE	5-6	2	binary	Date stamp: Julian year
RDATE	7-8	2	packed decimal	Date stamp: Julian day
RLEN	9	1	binary	Hex: 00=Variable data length

Entry Validation Record

This record is generated when the Network Access Manager (NAM) attempts to validate network access. If NAM is not active in the system, no recording is performed. Its length is 219 bytes.

Field Name	Offsets	Length	Format	Description
RTYPE	0	1	binary	Hex: 02=Entry validation record subtype
RTIME	1-4	4	binary	Time stamp (TOD clock)
RDATE	5-6	2	binary	Date stamp: Julian year
RDATE	7-8	2	packed decimal	Date stamp: Julian day
RLEN	9	1	binary	Hex: D1, Decimal: 209 Variable data length
RC	10	1	binary	Return code
ACTCP	11-18	8	character	Active control point name
USER	19-27	9	character	Length/user ID
GROUP	28-36	9	character	Length/group
ACCT	37-182	146	character	Count/length/account
PROC	183-191	9	character	Length/PROC
TERM	192-200	9	character	Length/TERM
APPL	201-209	9	character	Length/APPL
REQCP	210-218	9	character	Length/requested control point

(VSSULGN) CL/SUPERSESSION User Logon

This record is generated when a user establishes a session with CL/SUPERSESSION. Its length is 34 bytes.

Field Name	Offsets	Length	Format	Description
RTYPE	0	1	binary	Hex: E0, Decimal: 224 SS user logon subtype
RTIME	1-4	4	binary	Time stamp (TOD clock)
RDATE	5-6	2	binary	Date stamp: Julian year
RDATE	7-8	2	packed decimal	Date stamp: Julian day
RLEN	9	1	binary	Hex: 18, Decimal: 24 Variable data length
APPL	10-17	8	character	VTAM application name (left justified)
LU	18-25	8	character	VTAM logical unit (left justified)
USER	26-33	8	character	User ID (left justified)

(VSSULGF) CL/SUPERSESSION User Logoff

This record is generated when a user logs off or is disconnected, or when the control dialog is terminated. Its length is 34 bytes.

Field Name	Offsets	Length	Format	Description
RTYPE	0	1	binary	Hex: E1, Decimal: 225 SS user logoff subtype
RTIME	1-4	4	binary	Time stamp (TOD clock)
RDATE	5-6	2	binary	Date stamp: Julian year
RDATE	7-8	2	packed decimal	Date stamp: Julian day
RLEN	9	1	binary	Hex: 18, Decimal: 24 Variable data length
APPL	10-17	8	character	VTAM application name (left justified)
LU	18-25	8	character	VTAM logical unit (left justified)
USER	26-33	8	character	User ID (left justified)

(VSSTLGN) CL/SUPERSESSION Virtual Terminal Logon

This record is generated when a new virtual session is established. Its length is 38 bytes.

Field Name	Offsets	Length	Format	Description
RTYPE	0	1	binary	Hex: E2, Decimal: 226 SS terminal logon subtype
RTIME	1-4	4	binary	Time stamp (TOD clock)
RDATE	5-6	2	binary	Date stamp: Julian year
RDATE	7-8	2	packed decimal	Date stamp: Julian day
RLEN	9	1	binary	Hex: 1C, Decimal: 28 Variable data length
USER	10-17	8	character	User ID (left justified)
PLU	18-25	8	character	Virtual primary logical unit (left justified)
SLU	26-33	8	character	Virtual secondary logical unit (left justified)
LRN	34-37	4	binary	Session resource ID

For SMF statistics, session switches are determined by counting the number of times the virtual terminal buffer is REFRESHed. This means that when a REFRESH occurs, it is counted as a session switch. Hence, an IMBRCST saves the virtual terminal buffer and then REFRESHes it when the message is erased from the terminal. This creates an invalid session switch counter.

Remember that when you initially log on, the virtual terminal is not REFRESHed; it is created. So the LOGON count is increased, but the session switch count is not.

(VSSTLGF) CL/SUPERSESSION Virtual Terminal Logoff

This record is generated when a virtual session is terminated. Its length is 88 bytes.

Field Name	Offsets	Length	Format	Description
RTYPE	0	1	binary	Hex: E3, Decimal: 227 SS terminal logoff subtype
RTIME	1-4	4	binary	Time stamp (TOD clock)
RDATE	5-6	2	binary	Date stamp: Julian year
RDATE	7-8	2	packed decimal	Date stamp: Julian day
RLEN	9	1	binary	Hex: 4E, Decimal: 78 Variable data length
USER	10-17	8	character	User ID (left justified)
PLU	18-25	8	character	Virtual primary logical unit (left justified)

Field Name	Offsets	Length	Format	Description
SLU	26-33	8	character	Virtual secondary logical unit (left justified)
LRN	34-37	4	binary	Session resource ID
VLU	38-45	8	character	VTAM logical unit (left justified)
ACCLN	46-87	2	binary	Length of accumulator array is variable from 2-40 bytes and may include the following:
INMSG		4	binary	Inbound message count (APPL to SS)
INCNT		4	binary	Inbound byte count (APPL to SS)
OTCNT		4	binary	Outbound byte count (SS to APPL)
OTMSG		4	binary	Outbound message count (SS to APPL)
RFMSG		4	binary	Refresh message count (SS to TERM)
RFCNT		4	binary	Refresh byte count (SS to TERM)
UPMSG		4	binary	Update message count (SS to TERM)
UPCNT		4	binary	Update byte count (SS to TERM)
TIMSG		4	binary	Terminal input message count
TICNT		4	binary	Terminal input byte count

If data compression is on, you can calculate *outbound* compression efficiency by getting the absolute value of update byte count minus inbound byte count, multiplying by 100, and dividing by inbound byte count:

```
(ABS(UPCNT-INCNT)) * 100 / INCNT
```

and *inbound* compression efficiency by dividing outbound byte count by terminal input byte count:

```
OTCNT/TICNT
```

(GWLIMIT) CL/GATEWAY Record

This record is generated when the limit has been reached for resolving a data element.

Field Name	Offsets	Length	Format	Description
RTYPE	0	1	binary	Hex: F3, Decimal: 243 Gateway limit reached
APPL	1-8	8	character	VTAM application name (left justified)
LU	9-16	8	character	VTAM logical unit (left justified)
USER	17-24	8	character	User ID (left justified)
DATA	25-32	8	character	Data element (left justified)

(GWLOGON) CL/GATEWAY Logon

This record is generated when a user establishes a session with CL/GATEWAY. Its length is 34 bytes.

Field Name	Offsets	Length	Format	Description
RTYPE	0	1	binary	Hex: F0, Decimal: 240 Gateway logon
RTIME	1-4	4	binary	Time stamp (TOD clock)
RDATE	5-6	2	binary	Date stamp: Julian year
RDATE	7-8	2	packed decimal	Date stamp: Julian day
RLEN	9	1	binary	Hex: F3 Decimal:243 Gateway limit reached
APPL	10-17	8	character	VTAM application name (left justified)
LU	18-25	8	character	VTAM logical unit (left justified)
USER	26-33	8	character	User ID (left justified)

(GWLOGOFF) CL/GATEWAY Logoff

This record is generated when a user logs off CL/GATEWAY. Its length is 34 bytes.

Field Name	Offsets	Length	Format	Description
RTYPE	0	1	binary	Hex: F1, Decimal: 241 Gateway logoff
RTIME	1-4	4	binary	Time stamp (TOD clock)
RDATE	5-6	2	binary	Date stamp: Julian year
RDATE	7-8	2	packed decimal	Date stamp: Julian day
RLEN	9	1	binary	Hex: 18, Decimal: 24 Variable data length
APPL	10-17	8	character	VTAM application name (left justified)
LU	18-25	8	character	VTAM logical unit (left justified)
USER	26-33	8	character	User ID (left justified)

(GWPTSTRT) CL/GATEWAY SINGLE Start

This record indicates that a SINGLE session was started between the user and the selected destination application. Its length is 50 bytes.

Field Name	Offsets	Length	Format	Description
RTYPE	0	1	binary	Hex: F4, Decimal: 244 Gateway SINGLE start subtype
RTIME	1-4	4	binary	Time stamp (TOD clock)
RDATE	5-6	2	binary	Date stamp: Julian year
RDATE	7-8	2	packed decimal	Date stamp: Julian day
RLEN	9	1	binary	Hex: 28, Decimal: 40 Variable data length
APPL	10-17	8	character	VTAM application name (left justified)
LU	18-25	8	character	VTAM logical unit (left justified)
USER	26-33	8	character	User ID (left justified)
VLU	34-41	8	character	Virtual LU (left justified)
DEST	42-49	8	character	Destination LU (left justified)

(GWPTSTOP) CL/GATEWAY SINGLE Stop

This record indicates that a SINGLE session was terminated. Its length is 50 bytes.

Field Name	Offsets	Length	Format	Description
RTYPE	0	1	binary	Hex: F5, Decimal: 245 Gateway SINGLE stop subtype
RTIME	1-4	4	binary	Time stamp (TOD clock)
RDATE	5-6	2	binary	Date stamp: Julian year
RDATE	7-8	2	packed decimal	Date stamp: Julian day
RLEN	9	1	binary	Hex: 28, Decimal: 40 Variable data length
APPL	10-17	8	character	VTAM application name (left justified)
LU	18-25	8	character	VTAM logical unit (left justified)
USER	26-33	8	character	User ID (left justified)
VLU	34-41	8	character	Virtual logical unit (left justified)
DEST	42-49	8	character	VTAM destination LU (left justified)

Chapter 12. CT/Engine Startup Parameters

This chapter discusses:

- the location of CL/SUPERSESSION and CL/GATEWAY parameters to override
- considerations for calculating a value for the MINIMUM parameter
- considerations for using the QUIESCE parameter
- the CT/Engine initialization and customization parameters in alphabetic order

Locating Parameters to Override

Default startup parameter values for CL/SUPERSESSION and CL/GATEWAY are defined in member KLSSYSIN of *&rhilev*.RLSPARM. To override these parameters at initialization time, code your parameters in KLSSYSIN or the EXEC PARM field of the JCL. JCL parameters take precedence over KLSSYSIN.

If you want to override values for CL/SUPERSESSION or CL/GATEWAY, be sure to use member KLSSYSIN of the RLSPARM library, even though the parameters described below may refer to TLVSYSIN and TLVPARM.

Important

Changes to the startup parameters may adversely affect CT/Engine performance.

Calculating MINIMUM

If you want to use extended storage, you must code a non-zero value for the MINIMUM parameter, or allow MINIMUM to default. To determine the best value for MINIMUM, use the following formula:

```
2048K + (#users * ((#sess/user * screen_size * eab) + 20K)) * 1/8
```

2048K	Minimum storage CT/Engine must have to load.
#users	Number of simultaneous users.
#sess/user	Number of active sessions per user.

screen_size	Width of screen multiplied by length of screen. For example, a 3270 MOD2 terminal would have a **screen_size** value of (24 * 80) = 1920.
eab	One of the following: **1** No EAB support. **2** EAB support.
20K	Variable and Table storage.
1/8	Specifies that 1/8th of all users must be able to be logged on to CL/SUPERSESSION at the same time.

Note: Repeat **`(#sess/user * screen_size * eab)`** in the formula for each different screen size and EAB.

Approximately 32K per user allows either 3 EAB Model2 sessions or 6 non-EAB sessions. For Model4 and Model5, 42K per user allows either 3 EAB Model2 sessions or 6 non-EAB sessions.

For more information about the MINIMUM parameter, refer to the alphabetic list of CT/Engine startup parameters on the following pages.

Using QUIESCE

The QUIESCE limit has different affects on CARVED and FREE storage checking.

If you ignore CARVED storage checks (accomplished by coding QUIESCE(0,C) in your SYSIN member) an abend may occur without warning messages.

If you code a carved storage limit, once that percentage of storage has been carved, warning messages will alert you that QUIESCE Mode is in effect. This allows you to notify users and perform necessary shutdown procedures. Because CARVED QUIESCE mode is terminal, CL/SUPERSESSION will not honor new session requests regardless of how much storage is FREE. In CARVED QUIESCE mode an abend may still occur if a request for storage is made from users currently logged on that cannot be honored by the amount that was left reserved.

A FREE quiesce condition is recoverable. Storage allocated to user sessions is freed as users logoff active applications or complete dialog functions that require a virtual session. As soon as the amount of storage in use drops below the recovery point, the quiesce condition is relieved.

If you ignore FREE storage checks by coding QUIESCE(0,F) in your KLSSYSIN member and all of your allocated storage is in use when a request for additional storage is made, an abend condition will occur.

If you code a free storage limit, once that percentage of storage is in use, warning messages will alert you that quiesce mode is in effect. This condition will continue only as long as storage remains exhausted. Pre-Version 145 releases of CL/SUPERSESSION contained a FREE storage quiesce threshold of 88%.

You can minimize free storage exhaustion by preallocating the small block sizes into pools of all one size carved contiguously. See “Storage Preallocation” on page 214 for more information.

If either carved or free quiesce limit is reached, review storage totals and check MINIMUM and MAXIMUM values. After reviewing the storage totals you may need to increase MINIMUM or MAXIMUM values. Candle recommends that you allocate enough storage to include a small buffer above the quiesce threshold so that user processing is not interrupted.

For more information about the QUIESCE parameter, refer to the alphabetic list of CT/Engine startup parameters on the following pages.

CT/Engine Startup Parameters

CT/Engine startup parameters are defined with appropriate defaults in member TLVSYSIN of TLVPARM. CT/Engine startup parameters specify the execution and resource management environment.

You may override certain CT/Engine operating parameters at initialization time by coding the parameters in TLVSYSIN or the EXEC PARM field of the JCL. JCL specification PARMs override TLVSYSIN parameters.

AMODE31

```
AMODE31(Y | N)
```

Determines whether CT/Engine runs in 31-bit addressing mode. Y is the default for MVS/XA™ and MVS/ESA. N is the default for MVS/SP™.

If MVS/XA is running under VTAM prior to Version 3.1.1 or DFP prior to Version 1.2, code the following:

```
AMODE31(N)
```

APF

```
APF(Y | N)
```

Determines whether CT/Engine runs as APF-authorized. Y is the default if the CT/Engine job step is APF-authorized; otherwise, N is the default.

Note: If you specify Y, the CT/Engine job step must be APF-authorized.

| ***CONFIRM***

|

```
CONFIRM(15 | n)
```

| Sets the maximum number of seconds between two successive SHUTDOWN
| commands or MVS STOP (P) commands to terminate the CT/Engine address space.

| CONFIRM(0) allows CT/Engine shutdown to begin immediately without an
| additional, confirming SHUTDOWN command.

| CONFIRM(*n*) prevents accidental shutdowns by requiring you to confirm the
| command by entering it a second time within the specified number of seconds.

| For example, the default (*15*) requires you enter SHUTDOWN twice within 15
| seconds to terminate the CT/Engine address space.

| ***CONSECHO***

|

```
CONSECHO(N | Y)
```

| Determines whether CT/Engine command output is routed back to the originating
| MVS operator console.

| *N* causes all output from commands issued via the MVS MODIFY command or
| subsystem consoles to be routed only to the master console and those consoles
| which are receiving the appropriate MVS route codes.

| *Y* causes the command output to be routed specifically to the originating console, as
| well as the master and other consoles. By coding *Y*, products such as NetView and
| AF/OPERATOR are able to review the output of CT/Engine commands.

| ***DATEFMT***

|

```
DATEFMT(MMDDYY | DDMMYY | YYMMDD)
```

| Specifies the format of dates displayed by CT/Engine in the &SYSDATE dialog
| variable. Possible values are:

MMDDYY	U.S. format
DDMMYY	European format
YYMMDD	International format

| ***DEBUG***

Important

Do not modify this parameter except under the guidance of a Candle support representative.

```
DEBUG(N | Y)
```

Specify Y to use the debug option for resolving internal problems.

DEBUG(Y) automatically turns on the internal trace with the default number of entries (1024). See "TRACE" on page 202 for more information.

DEBUG(Y) will cause an increase in CPU and storage use, as well as the issuance of more messages. The amount of the increase depends on the activity within the address space.

INBDLIM

```
INBDLIM(25 | n)
```

Specifies the maximum number of messages that can be queued for a session between an application and CT/Engine. If this number is exceeded, the session terminates.

INBDLIM prevents applications from flooding CT/Engine with messages, which causes CT/Engine to run short of storage and abend. By specifying a large value, an application that sends many outbound datastreams to a terminal without requesting input can cause CT/Engine to abend, even if definite response is not specified for the session.

Most file transfer programs implement handshaking protocols that require them to perform read operations. These programs should not be affected by a small INBDLIM value.

Note: INBDLIM limits the number of messages an application sends to CT/Engine (the virtual terminal). OUTBDLIM limits the number of messages CT/Engine sends to the physical terminal.

INBOUND

```
INBOUND(248 | n)
```

Specifies the size of all VTAM RECEIVE buffers, regardless of origin.

If the length of a received RU is larger than the value assigned to INBOUND, excess path lengths result. If the value assigned to INBOUND is much larger than the length of the RUs received, storage problems may result.

If you are running above VTAM 1.3, do not modify the INBOUND parameter.

INITIAL

```
INITIAL(membername)
```

Identifies the command list (CLIST) that contains CT/Engine initialization commands. There is no hard-coded default. TLVPARM(TLVSYSIN) contains INITIAL(KLVSTART), which invokes the KLVSTART member of TLVCMDS.

TLVCMDS may contain a member named *smfid*, where *smfid* is the SMF ID of the CPU executing CT/Engine. If this member exists, it is automatically invoked before the command list specified in the INITIAL parameter.

INITLIST

```
INITLIST(membername)
```

Identifies the member of TLVPARM which contains initialization member name overrides. Normally, CT/Engine initialization modules read members of TLVPARM whose names are the same as the modules. The member identified by the INITLIST keyword contains statements such as:

```
startup-module-name=override-name
```

Each line of the INITLIST member names a particular startup module and identifies the member of TLVPARM which should be read by that module for its parameters. The valid startup module names vary with the CT/Engine-based products that are installed. The number of startup modules may also change as new features are added or new products become available. As a result, it is not possible to provide a comprehensive list of startup module names.

No wild-card characters are accepted during INITLIST processing. Each override must be coded in full. Override statements must be coded one per line. The statements may appear in any column. Input will be converted to upper case prior to processing. Comment lines are indicated by an asterisk as the first non-blank character. On any line, anything following an asterisk is ignored.

Here is a sample input line:

```
KLVINNAM=NAMINIT0     *  Override KLVINNAM parameter member name
```

If multiple override statements for the same startup module are present, the last one will determine the member name to be used. No messages will be issued.

TLVPARM may contain a member named *smfid*, where *smfid* is the SMF ID of the CPU executing CT/Engine. If this member exists, it will be used as the list of initialization member name overrides if the INITLIST keyword is not coded. When INITLIST is coded, a member of TLVPARM named *smfid* will not be read even if present in the library.

INTLCHAR

```
INTLCHAR(N | Y)
```

Implements the following national language support features:

1. The window vertical bar character (|) is changed from X'6A' to X'4F'.
2. Characters with diacritical marks that were previously interpreted as invalid are now valid.

LIMIT

Important

Do not modify this parameter except under the guidance of a Candle support representative.

```
LIMIT(16,P | n,P)
LIMIT(16,X | n,X)
```

Specifies the largest block of primary (*P*) or extended (*X*) storage that can be allocated. This value is specified in bytes, as a power of 2. For example, if *n* is 16, the largest block that can be allocated is 65,536 bytes.

Primary storage is below the 16-megabyte line; extended storage is above the line. To specify values for both primary and extended storage, include the LIMIT parameter twice in TLVSYSIN:

```
LIMIT(n,X)
LIMIT(n,P)
```

If the LIMIT value is too small and a process in CT/Engine attempts to allocate a block of storage larger than LIMIT specifies, program interruption U0100 or U0200 results. Too large a LIMIT value may waste storage and increase processing overhead.

LOGBLOCK

```
LOGBLOCK(3120 | n)
```

Specifies, in bytes, the block size of the CT/Engine TLVLOG file.

LOGBUFS

Important

Do not modify this parameter except under the guidance of a Candle support representative.

```
LOGBUFS(2 | n)
```

Specifies the number of buffers to allocate for the CT/Engine TLVLOG file.

If the value of LOGBUFS is small and extensive logging is performed (for example, during debugging), CT/Engine response may suffer because of excessive physical I/O. If the value of LOGBUFS is large, storage shortages may occur.

LSRPOOL

```
LSRPOOL(size,count[,0 | hiper])
```

No default; however, you must code at least one LSRPOOL parameter in order for CT/Engine to start.

Corresponds to the BUFFERS parameter of the BLDVRP macro instruction, and specifies the number of buffers to be made available for each VSAM dataset used by CT/Engine.

size is a buffer (VSAM control interval) size. Valid sizes are 512, 1024, 2048, 4096, 8192, 12288, 16384, 20480, 24576, 28672, and 32768.

count is the number of virtual storage buffers of *size* to be allocated. The minimum is 3. The maximum is 65535, although this may be less, depending on the amount of available virtual storage in the CT/Engine address space. Buffers will be allocated from extended storage if AMODE31(Y) was coded or defaulted to; from primary storage otherwise.

hiper is the number of hiperspace buffers of *size* to be allocated. The minimum is 0 (no hiperspace buffers). The maximum is 16777215, although this may be less, depending on the hiperspace storage available to IBM VSAM services. Ensure the CT/Engine address space is non-swappable when you are using hiperspace buffers.

Note: IBM restricts hiperspace buffers to multiples of 4K. Do not code *hiper* for the 512, 1024, or 2048 sizes; an error message will be issued and CT/Engine startup will terminate.

For best storage use, code an LSRPOOL parameter for each different VSAM control interval size that CT/Engine uses:

- index buffer for NAM and table database
- data buffer for NAM and table database
- data buffer for VIEWLOG

Candle distributes the recommended values in TLVPARM(TLVSYSIN).

Note: You must enter LSRPOOLs individually; you cannot string them.

If you receive many KLVVS026 messages identifying buffer contention, increase the number of buffers allocated to the dataset identified in the associated KLVVS021 messages.

LSRSTRNO

```
LSRSTRNO(32 | n)
```

Corresponds to the STRNO parameter of the BLDVRP macro instruction. It is the maximum number of concurrent VSAM requests that CT/Engine can process against all the VSAM datasets allocated to it.

If you receive many KLVVS026 messages identifying string contention, or if the STRMAX value in the KLVVS002 messages issued during CL/ENGINE shutdown is consistently the same as the value in TLVSYSIN, increase the LSRSTRNO value.

The minimum value is 1. The maximum is 255.

MAXIMUM

```
MAXIMUM(8192,P | n,P)
MAXIMUM(n,X)
```

The default value for extended storage (*X*) is the value specified by the MINIMUM keyword.

MAXIMUM is a storage throttle used to prevent GETMAINs from overallocating and occupying the page dataset with rarely referenced frames.

The variable *n* represents the maximum amount (in kilobytes) of primary or extended storage that can be allocated. *X* stands for extended storage (above the 16-megabyte line), and *P* stands for primary storage (below the line).

Set your MAXIMUM value to a value that will allow CT/Engine to continue running without overloading your page volumes when the steady-state MINIMUM value is exceeded.

To use extended storage, you must do both of the following:

- Code the MINIMUM parameter.
- Make sure that MAXIMUM is equal to or greater than MINIMUM + RESERVE.

  If MAXIMUM is too large and RESERVE is not large enough to meet your requirements, the address space may run out of virtual storage.

If the value of MAXIMUM is greater than that of MINIMUM, CT/Engine attempts a conditional GETMAIN for the MAXIMUM value minus the RESERVE value

(RESERVE defaults to 512,P.). If the MAXIMUM value is not satisfied, CT/Engine accepts the amount of storage acquired by the GETMAIN.

MINIMUM

```
MINIMUM(1024,P | n,P)
MINIMUM(8192,X | n,X)
```

n represents the mimimum amount (in kilobytes) of primary or extended storage that can be allocated.

For example, to specify a 16-megabyte above-the-line region, code

```
MINIMUM(16384,X)
```

To specify a 32-megabyte above-the-line region, code

```
MINIMUM(32768,X)
```

To use extended storage, you must do both of the following:

- Code the MINIMUM parameter.
- Make sure that MINIMUM + RESERVE is less than or equal to MAXIMUM.

Note the following about the default above-the-line region:

- Specified in the IEFUSI and IEALIMIT MVS modules.
- Distributed by IBM as 32 megabytes.
- If smaller than the amount specified for the MINIMUM parameter, do one of the following:
  - Alter the default.
  - Use the REGION parameter as follows:

0K or 0M	All primary and extended storage is available for GETMAIN.
Up to 16M	Primary region equals the specified value; extended region equals the default.
Up to 32M	All available region goes to primary storage; extended region equals the default.
Above 32M	All available region goes to primary storage; specified value goes to extended storage.

In general, Candle recommends REGION=0M.

OPLIMIT

```
OPLIMIT(0 | n)
```

Specifies the maximum number of characters that can be queued to a single CT/Engine operator before the messages are bypassed. The default (*0*) is no limit.

OPLOCAL

```
OPLOCAL(REPLY,ERROR | messagetype,messagetype,...)
```

Specifies the types of messages an operator console receives in response to actions initiated by that same console.

With the default OPLOCAL setting, an operator performing an action that generates a CT/Engine ALERT message does not receive the message at the console unless OPMASK allows it. For information on message types, see *CL/GATEWAY for MVS and CL/GATEWAY for IMS Messages*.

OPMASK

```
OPMASK(INFO,WARN,ALERT | messagetype,messagetype,...)
```

Specifies the types of unsolicited messages all operator consoles receive. For information on message types, see *CL/GATEWAY for MVS and CL/GATEWAY for IMS Messages*.

OPSTART

```
OPSTART(command)
```

Specifies an initial CT/Engine command or CLIST to be issued after an operator logs on. There is no default. TLVPARM(TLVSYSIN) contains OPSTART(KLVOPST), which invokes the KLVOPST member of TLVCMDS.

OUTBDLIM

```
OUTBDLIM(500 | n)
```

Specifies the maximum number of messages that can be queued for a session between CT/Engine and the physical terminal. If this number is exceeded, the session terminates.

The purpose of OUTBDLIM is to prevent excessive messages from causing CT/Engine to run short of storage and abend. If you specify a very large value for OUTBDLIM, an application that sends many outbound datastreams to a terminal without requesting input can cause CT/Engine to abend, even if definite response is not specified for the session.

Note: OUTBDLIM limits the number of messages CT/Engine sends to the physical terminal. INBDLIM limits the number of messages an application sends to CT/Engine (the virtual terminal).

OUTBOUND

```
OUTBOUND(504 | n)
```

Specifies, in bytes, an outbound RU buffer length for VTAM sessions for which a length is not provided in the session parameters.

PACK

Important

Do not code this parameter except under the guidance of a Candle support representative.

```
PACK(Y | N)
```

Determines how CT/Engine loads modules into the CT/Engine region:

omitted	Specifies that modules are placed into the next available doubleword storage location. If the address space is APF-authorized, an MVS directed load is used and MVS CDEs are available; otherwise CT/Engine performs a pseudo-directed load and no CDEs are created.
Y	CT/Engine performs a pseudo-directed load to place modules into the next available doublword storage location. No MVS CDEs are created.
N	Modules are loaded using standard MVS LOAD macros. MVS CDEs are created. Most CT/Engine-based modules are linkedited on a page boundary; PACK(N) honors that boundary, and therefore will require a larger CT/Engine region.

QUIESCE

```
QUIESCE(90,P,F | n,P,F)
QUIESCE(95,P,C | n,P,C)
QUIESCE(90,X,F | n,X,F)
QUIESCE(95,X,C | n,X,C)
```

Sets thresholds for slowing down allocation of storage. When the thresholds are exceeded, quiesce mode goes into effect, causing rejection of all conditional storage requests. Conditional storage requests include session and dialog startup, as well as other functions that can recover from a storage shortage. Unconditional requests for storage continue to be satisfied in quiesce mode.

The variable *n* represents the percentage of primary (P) or extended (X), free (F) or carved (C) storage allowed to be allocated before quiesce mode takes effect. Carved storage is storage put into use for the first time and allocated to a specific size; free storage is storage that has been carved but is not in use. Primary storage is below the 16-megabyte line; extended storage is above the line.

A value of zero (0) indicates no QUIESCE threshold, which means quiesce mode never goes into effect. A value of 100 indicates that all storage must be used or carved before quiesce mode takes effect.

To specify that quiesce mode takes effect when 80% of the total amount of storage is in use in extended storage, enter the following:

```
QUIESCE(80,X,F)
```

To specify that quiesce mode takes effect when 90% of the total amount of primary storage has been carved, enter the following:

```
QUIESCE(90,P,C)
```

When CT/Engine detects that the amount of storage carved or in use is above the QUIESCE threshold percentage, it enters quiesce mode and does not permit new dialogs or sessions to start. For free storage, quiesce mode continues until the amount of storage in use drops below a recovery point, calculated as follows:

```
Recovery point = Threshold amount - ((MAXIMUM - Threshold amount) / 2)
```

For example, with a MAXIMUM of 10 megabytes and a QUIESCE value of 90:

```
Recovery point = 9M - ((10M - 9M) / 2) = 8.5M
```

For carved storage, quiesce mode has no recovery point. Until CT/Engine is recycled, no new sessions or dialogs can be started. However, existing sessions and dialogs continue to operate normally.

You can use the STGMON parameter to control reporting on quiesce mode conditions. See "STGMON" on page 201.

RESERVE

```
RESERVE(512,P | n,P)
RESERVE(2048,X | n,X)
```

The variable *n* represents the number of kilobytes of primary (P) or extended (X) storage to set aside for other routines that may perform their own GETMAINs in the CT/Engine address space (for example, ACF2 and RACF). The default for primary storage (P) is 512. The default for extended storage (X) is 2048. Primary storage is below the 16-megabyte line, and extended storage is above the line.

To specify values for both primary and extended storage, include the RESERVE parameter twice in TLVSYSIN:

```
RESERVE(n,X)
RESERVE(n,P)
```

If the RESERVE value is larger than the MINIMUM value, CT/Engine terminates. The total of the MINIMUM and RESERVE values must be less than or equal to the MAXIMUM value.

If you have ACF2 or RACF installed, the default setting (512,P) supports about 500 users. ACF2 and RACF use approximately 1K of primary storage per logged-on user. If your RESERVE value is too small, you may encounter IST566I messages from VTAM or S80A, S878, S066, S40D, or S0F9 abends.

SDUMP

```
SDUMP(Y | N | S | M)
```

Determines whether or not SVC dumps are generated. If the CT/Engine job step is APF-authorized, the default is Y. Otherwise, the default is N, and you must APF-authorize the CT/Engine job step before you can specify SDUMP(Y).

Y Specifies that an SVC dump is to be directed to a system dump dataset (SYS1.DUMP*xx*).

To capture an entire SVC dump, perform these steps:

1. Ensure that the SYS1.DUMP*xx* datasets are large enough to hold the contents of the CT/Engine address space.
2. Determine the size of the CT/Engine address space.
3. Provide enough DASD in the SYS1.DUMP*xx* datasets to accommodate the SDUMP requirements.

For MVS/ESA 3.1.3 and higher, SDUMP writes an unblocked 4160-byte record for each page of virtual storage being dumped. For earlier versions of MVS, the record size is 4104 bytes.

Candle recommends these storage guidelines for SVC DUMP datasets:

- Allocate 32 meg if 16 to 32 meg are specified in TLVSYSIN for MINIMUM/MAXIMUM keywords.
- For each additional 16 meg specified in TLVSYSIN for MINIMUM/MAXIMUM keywords, add 25 meg for the SVC DUMP dataset.

For example:

```
MINIMUM (16384,x) - Allocate a 32 meg SVC DUMP dataset
MINIMUM (32768,x) - Allocate a 57 meg SVC DUMP dataset
MINIMUM (49152,x) - Allocate a 82 meg SVC DUMP dataset
MINIMUM (65536,x) - Allocate a 107 meg SVC DUMP dataset
```

N
Specifies that a formatted dump should be directed to the TLVSNAP dataset. Formatted dumps should be avoided when possible; they disable the CT/Engine address space for a longer period of time than either SVC dumps or SYSMDUMPs, and are more difficult to analyze.

S
Specifies that a summary dump only should be directed to the TLVSNAP dataset. A summary dump consists of an abend summary and a dispatcher summary only. **Warning:** Summary dumps do not provide enough information to allow problems to be analyzed. SDUMP(S) is provided only for very specific testing purposes, where it is known that a dump will not be needed.

M
Specifies that the system should take an ABEND dump to a dataset defined with the SYSMDUMP ddname. This type of dump is not formatted by the operating system and must be analyzed with IPCS. Only the first dump taken will be captured on the SYSMDUMP dataset *unless* the JCL specifies DISP=MOD, in which case multiple dumps may be collected on the same dataset.

CT/Engine automatically initializes the SYSMDUMP dataset with an end-of-file mark at initialization. If DISP=SHR or DISP=OLD is specified for the SYSMDUMP dataset, existing dumps will be overwritten. If DISP=MOD is specified, the system will write the dump following any previous dumps.

If SDUMP(M) is specified and the SYSMDUMP DDNAME is missing, or the initialization fails, startup is aborted.

Note: The CT/Engine address space need not be APF-authorized to take a SYSMDUMP. For more information regarding SYSMDUMP specification, refer to IBM's *Planning: Problem Determination and Recovery*.

STGMON

```
STGMON(15 | n)
```

Specifies the number of minutes between storage quiesce mode message displays. Any value between 0 and 120 is valid. A value of 0 results in messages being issued only when a short-on-storage condition is detected or relieved (for example, when the quiesce mode state changes).

SWAP

```
SWAP(N | Y)
```

Specifies whether the CT/Engine job step is APF-authorized and the CT/Engine region is swappable. The default is N if the CT/Engine job step is APF-authorized. Otherwise, it is Y.

TASKS

Important

Do not modify this parameter except under the guidance of a Candle support representative.

```
TASKS(n)
```

Default is the number of available processors.

Specifies the number of general-purpose subtasks to be attached in the CT/Engine address space. If CT/Engine is running on a multiprocessor, the TASKS default increases both throughput and CPU usage. Reducing the number of tasks decreases both throughput and CPU usage.

In general, reduce the value of TASKS only when CPU usage is a concern and system paging is low, since fewer tasks will be available for performing work whenever other tasks are in a page-fault wait.

TRACE

Important

Do not modify this parameter except under the guidance of a Candle support representative.

```
TRACE(11 | n)
```

Specifies, as the exponent of 2, the number of internal trace table entries to reserve. Each entry consists of 32 bytes. For example, *TRACE(12)* reserves 4096 trace entries.

If TRACE is not specified, no internal trace table will be allocated *unless* DEBUG(Y) is specified, in which case a default trace value of 10 is used, reserving 1024 trace table entries.

The internal trace table is included in a dump and provides useful diagnostic information for Candle Support Services. Elements that are included in the internal trace table are controlled by the CT/Engine TRACE operator command. Contact Candle Support Services for information about this command.

UPPERDLG

```
UPPERDLG(N | Y)
```

Determines whether or not SSPL dialog output is folded to upper case before displaying on users' terminals.

Candle recommends that you specify Y only if your users' terminals display Kanji or other special characters in place of lowercase characters.

UPPERLOG

```
UPPERLOG(N | Y)
```

Determines whether or not output from the LOG SSPL dialog function is folded to upper case. (LOG output is written to TLVLOG.)

Candle recommends that you specify Y only if your terminals display Kanji or other special characters in place of lowercase characters.

UPPERWTO

```
UPPERWTO(N | Y)
```

Determines whether or not output from the WTO SSPL dialog function is folded to upper case. (WTO output is written to the MVS consoles.)

Candle recommends that you specify Y only if your MVS consoles display Kanji or other special characters in place of lowercase characters.

WTO

```
WTO(Y | N)
```

Determines whether or not CT/Engine issues WTOs.

WTOs write information and exception condition messages to the operator consoles. ALERT messages are always written to the consoles.

Note: WTO(N) will suppress messages written with the WTO SSPL dialog function.

WTODC

```
WTODC(ALERT,2 | type,code,code,...)
```

Specifies WTO descriptor codes for CT/Engine message types.

Specify one WTODC parameter for each CT/Engine message type. For example, to assign descriptor code 7 (Application Program/Processor) to CT/Engine error messages, enter the following:

```
WTODC(ERROR,7)
```

For definitions of the descriptor codes, see IBM's *Supervisor Services and Macro Instructions* manual. For information on message types, see *CL/GATEWAY for MVS and CL/GATEWAY for IMS Messages*.

WTORC

```
WTORC(ALERT,1,8,11 | type,code,code,...)
```

Specifies WTO route codes for for CT/Engine message types. Specify the WTORC parameter for each CT/Engine message type.

For definitions of the route codes, see IBM's *Supervisor Services and Macro Instructions* manual. For information on message types, see *CL/GATEWAY for MVS and CL/GATEWAY for IMS Messages*.

Chapter 13. Response Time Monitor Interface

CL/SUPERSESSION provides an interface to the Candle End-to-End Response Time Feature™ (ETE), the IBM NetView® Performance Monitor (NPM) Synergy Interface (NSI), and the Legent NetSpy™ response time monitor (RTM).

There are two types of sessions created when response time monitors are used.

control session	A session between the physical terminal LU and the CL/SUPERSESSION entry point ACB. You are in session with CL/SUPERSESSION when you are under dialog control. For example, when you are: ● doing administrative work from the main menu ● invoking a dialog through a trigger
relay session	A session that is created when you jump into an application such as TSO or CICS. A relay session is made up of two real sessions: one between the physical terminal and CL/SUPERSESSION entry point ACB and one between a virtual terminal and the application.

Response time monitoring products need to know when you create a new relay session and when you are in a control session. This is done by mapping messages provided by CL/SUPERSESSION to the response time monitor every time a session type switch occurs, or when jumping from one relay session to another. Response time monitoring products then use the mapping messages to correlate statistics to the correct session.

Support for NPM and ETE (or NetSpy) can be activated independently or concurrently. This chapter describes the procedures for installing these products.

The CT/Engine RTM command activates the interface to ETE, NPM, and NetSpy. For more information on the RTM operator command, see the *Operator's Guide*.

NPM Recommended Installation Procedure

To use the RTM support for NPM, you must do the following:

- use MVS/XA or MVS/ESA
- install NPM version 1.4 or higher
- install CL/SUPERSESSION at the same host where NPM/NSI resides

1. Assemble and link edit the exit, *&rhilev*.RLSSAMP(KLSXNPM), and put the resulting load module in the CT/Engine load library (*&rhilev*.RLSLOAD). With the USEREXIT you can control which 8 bytes of the 40-byte ACCOUNT CODE field provided by CL/SUPERSESSION are passed to NPM/NSI.

Note: NPM/NSI allows for 8-byte ACCOUNT CODE field. If no user exit is specified, CL/SUPERSESSION passes the first 8 bytes of the ACCOUNT CODE field to NPM/NSI by default.

2. Copy module FNMNSI to *&rhilev*.RLSLOAD library, or concatenate the NPM load library to *&rhilev*.RLSLOAD.
3. Authorize *all* concatenated *&rhilev*.RLSLOAD libraries.
4. Define the CL/SUPERSESSION VTAM application name in the NPM initialization member FNMINIT, specifying `TYPE=NVAS` on the APPL statement.
5. Define applications to be monitored (for example CICS and TSO) with APPL statements in the NPM initialization member FNMINIT.

   Note: For VTAM applications such as TSO and NETVIEW, where the VTAM ACB name is different from the application name, NPM/NSI needs both ACBNAME and APPL name to correctly correlate data traffic to the appropriate session.

   For example, if your TSO definitions in SYS1.VTAMLST are as follows:

```
TSOF     APPL   AUTH=...,ACBNAME=TSO,EAS=1
TSOF0001 APPL   AUTH=...,ACBNAME=TSO0001,EAS=1
TSOF0002 APPL   AUTH=...,ACBNAME=TSO0002,EAS=1
TSOF0003 APPL   AUTH=...,ACBNAME=TSO0003,EAS=1
  .
  .
  .
TSOF050 APPL   AUTH=...,ACBNAME=TSO0050,EAS=1
```

   You will need the following APPL statements in member FNMINIT:

```
APPL   NAME=TSOF,SYNMASK=TSO0001,SYNCOUNT=50
APPL   NAME=TSOF,SYNMASK=TSOF0001,SYNCOUNT=50
```

6. Refer to the IBM manual *NPM Installation and Customization* for more details.
7. Use the administrator functions to change the setting for RTM interface to Y in the common segment of the global profile.

   If you want to restrict the NPM interface to a selected group of users, change the setting for RTM interface to Y only in the common segment of the selected group profile.
8. Choose one of the following options:
   a. To start the interface automatically whenever CL/SUPERSESSION is initialized, add the command

   ```
   RTM ON TYPE=NPM USEREXIT=EXITNPM
   ```

   to *&rhilev*.RLSCMDS(KLSCINSS), and save it.

   b. To start the interface manually, issue the command

   ```
   RTM ON  TYPE=NPM USEREXIT=EXITNPM
   ```

   from the CT/Engine operator screen.

9. Use the \o trigger to invoke the KLSVTOPT dialog. A panel displays virtual session options. Verify that the RTM interface is active for your sessions. If necessary, change the options.

NPM Verifying the Installation

After installing the NPM/NSI interface in CL/SUPERSESSION and customizing NPM/NSI, verify the installation as follows:

1. Start the NSI address space.
2. Start the NPM address space. Look for message:

   ```
   FNM911I NPM (jobname) CONNECTED SUCCESFULLY WITH NSI
   ```

3. Issue the following CT/Engine operator command:

   ```
   RTM ON  TYPE=NPM USEREXIT=EXITNPM
   ```

4. From the DATA FILE AND MANAGEMENT panel in NPM verify that an FNMSESx file is active and not full.
5. Start SESSION collection in NPM, for both CL/SUPERSESSION and an application defined in FNMINIT.
   a. From the PRIMARY OPTIONS panel select option 1 (Data Collection).
   b. From the DATA COLLECTION OPTIONS panel select option 4 (Session Start).
   c. From the START SESSION panel enter the CL/SUPERSESSION entry point ACB in the “Resource name” row, and fill in the Start and Stop times.
   d. Repeat steps a-c for the selected application.
6. Logon to CL/SUPERSESSION and from the main menu select an application that is being monitored by NPM.
7. Issue the \o trigger to verify that RTM is on.
8. From NPM display session data for your physical terminal LU.
   a. From the PRIMARY OPTIONS panel select option 2 (Data Analysis).
   b. From the DATA ANALYSIS SELECTION panel select option 2 (NPM LU Detail Analysis).
   c. From the NPM LU DETAIL ANALYSIS panel select option 1 (Transit Time/Interval), enter the physical terminal LU name, and the FROM - TO date and time fields.

   NPM reports EXTENDED session analysis with userid and virtual terminal name under the SESSION MGR, USERID/SLU column. CONTROL session analysis displays only the USERID under the same column since there is no virtual session involved when a user is returned to CL/SUPERSESSION.

ETE (NetSpy) Recommended Installation Procedure

To use the RTM support for ETE/NetSpy, you must do the following:

- Install VTAM Version 3.1 or higher.
- Install CL/SUPERSESSION at the same host where ETE/NetSpy resides.

1. Assemble and link edit the exit, *&rhilev*.RLSSAMP(KLS XRTM), and put the resulting load module in the CT/Engine load library (*&rhilev*.RLSLOAD). The USEREXIT is required by the NetSpy interface but is optional for ETE. See "Customizing USEREXIT (KLSXRTM)" on page 210 for details.
2. Use the administrator functions to change the setting for RTM interface to Y in the common segment of the global profile.

   If you want to restrict the ETE/NetSpy interface to a selected group of users, change the setting for RTM interface to Y only in the common segment of the selected group profile.
3. Choose one of the following options:
   a. To start the interface automatically whenever CL/SUPERSESSION is initialized, add one of the following commands to *&rhilev*.RLSCMDS(KLSCINSS), and save it.

      ```
      RTM ON TYPE=ETE USEREXIT=EXITRTM     (ETE)

      RTM ON USEREXIT=EXITRTM             (NetSpy)
      ```
   b. To start the interface manually, issue one of the following commands from the CT/Engine operator screen.

      ```
      RTM ON TYPE=ETE USEREXIT=EXITRTM     (ETE)

      RTM ON  USEREXIT=EXITRTM            (NetSpy)
      ```
4. Use the \o trigger to invoke the KLSVTOPT dialog. A panel displays virtual session options. Verify that the RTM interface is active for your sessions. If necessary, change the options.
5. For NetSpy only, follow the directions specified in the NetSpy control library as defined by member NSYSMGRI (NetSpy Session Manager Install). Be sure to specify APPLNAME=NETNAME before any NetSpy APPL statements.
6. For NetSpy only, define CL/SUPERSESSION to NetSpy in the NetSpy INITPRM member:

   ```
   APPL=netname SMANAGER=CLSS FORCEDR=100
   ```

   where *netname* is the network name of the entry point CL/SUPERSESSION ACB.

Mapping Message Format

The mapping message has the following format:

Position	
0-6	Necessary 3270 control characters to write the message to the terminal screen.
7-13	This field is the eyecatcher header that identifies the datastream as a mapping message to ETE/NetSpy. The default is MONITOR. NetSpy requires `NETSPY@` in this field. ETE expects the default but also recognizes `NETSPY@`. To change the default, you must invoke the USEREXIT.
14-21	The new application name when a session type switch occurs, or when a user jumps from one relay session to another. For control sessions, this field contains the session manager name recognized by ETE/NetSpy. The default name is CLSS. You can override the default when activating RTM through the *smanager* positional parameter. ETE accepts the default. NetSpy requires that this name be the same as the SMANAGER= parameter on the NetSpy APPL statement. For relay sessions, this field contains the VTAM application name for the application that is brought to the foreground.
22-29	For relay sessions this field contains the virtual terminal's network address (subarea, element address). This field is blank when a control session mapping message is passed to ETE/NetSpy.
30-n	Any user data.

The mapping message is passed by CL/SUPERSESSION to the USEREXIT before being transmitted to the physical terminal. See “Customizing USEREXIT (KLSXRTM)” on page 210 for details.

Activating the interface

The RTM command activates the interface to ETE or NetSpy response time monitor. It allows you to specify the session manager name recognized by ETE and NetSpy and to activate a user exit. The syntax of the command is:

```
RTM  ON|OFF [smanager|CLSS] USEREXIT=exitname
```

ON	Activates the interface. If the interface is already active, it will be deactivated (as though an RTM OFF command were issued) before the activation request is processed.
OFF	Deactivates the interface, and discards the existing session manager name and user exit name. You will need to respecify these when you reactivate the interface.
smanager	Identifies the session manager name to ETE/NetSpy.
exitname	Identifies the user exit that inspects and/or modifies the mapping message just before it is transmitted to the physical terminal. NetSpy requires that you specify **USEREXIT=EXITRTM**. Member KLSXRTM in *&rhilev*.RLSSAMP contains instructions for implementing and extending the sample exit. Also, see “Customizing USEREXIT (KLSXRTM)” for details.

Customizing USEREXIT (KLSXRTM)

USEREXIT is required by NetSpy. It is optional for ETE .

This routine is used as a model for writing an RTM interface exit that can be used to mofidy the mapping message. This routine can be used as is to provide the interface to NetSpy. Upon invocation of this routine the following occurs:

- R15 contains the address of the module entry point. The first word of the USEREXIT must be a NOP instruction that specifies the length of any work area that will be allocated by CL/SUPERSESSION before the exit is invoked.
- R14 contains the return address in the calling module.
- R1 contains the address of the mapping message that will be transmitted to the physical terminal. This messagge is mapped by the following DSECT.

```
RTMMSG    DSECT                         RTM mapping message
RTMLL     DC   AL2(L'RTMTEXT)           Length of mapping message
RTMSTART  DC   XL7'F1401140401D6C'      3270 controls
RTMMON    DC   CL7'MONITOR'             RTM name recognized by RTM
RTMAPPL   DC   CL8'APPLNAME'            RTM application name
RTMSUB    DC   CL4'0001'                Virtual terminal's subarea address
RTMELE    DC   CL4'0001'                Virtual terminal's element address
RTMNETA   EQU  RTMSUB,*-RTMSUB          Virtual terminal's network address
RTMTEXT   EQU  RTMSTART,*-RTMSTART      Datastream as provided by caller
RTMWORK   DS   256X                     User area.
```

- R0 contains the address of the save area (and any additional work area that was requested) for this exit routine.
- R2 contains the address of the USER information block. This control block is READ-only and is mapped by the following DSECT.

```
RTMUSER  DSECT          User information block
RTUSERID DS    CL8      USERID
RTGROUP  DS    CL8      GROUPID
RTACCT   DS    CL40     ACCOUNTING information
RTPTERM  DS    CL8      Physical terminal LU name
```

The mapping datastream uses the 3270 WRITE command to eliminate the perceived flashing of the physical terminal screen that users experience with long response times. The mapping message is sent to the terminal after it is examined by ETE/NetSpy and must reside in a separate PIU from any regular output generated by CL/SUPERSESSION that follows the mapping message.

Because of device limitations in handling the ERASE WRITE or ERASE WRITE ALTERNATE command which is included in the regular output that follows the mapping message, CL/SUPERSESSION provides a feature that avoids the ERASE WRITE effect when outbound compression is on.

Note: This feature does not apply to sessions with outbound compression turned off and is provided with the following limitation.

If you modify the mapping message, you are limited to write to the physical screen buffer from row 1, col 1 (position 0), up to the length of the message as provided to you in the USEREXIT (up to position 285 of the terminal buffer). If you decide to append any information to the mapping message (for example, information provided to you in the User Information Block), you must update the mapping message length (RTMLL). The number of mapping message characters transmitted by CL/SUPERSESSION to the terminal is specified in the RTMLL field.

The results are unpredictable if you attempt to write to the screen beyond postion 285.

To turn off this feature you must comment out the following instruction in the sample USEREXIT.

```
OI     RTMLL,X'80'     TURN ON VALIDATION SWITCH
```

For ETE, the USEREXIT is invoked only to avoid the ERASE WRITE effect. To customize USEREXIT for ETE you can comment out the following instruction:

```
MVC    RTMMON,=CL7'NETSPY@'  Change monitor name to NetSpy
```

The contents of the eyecatcher datastream that is transmitted on the physical terminal session for external response time monitors works properly even with 3270 display devices in 16-bit addressing mode.

The eyecatcher datastream contains three 3270 order sequences that specify buffer address zero (row 1 column 1 of the display screen). This address is transmitted either in 12-bit addressing mode (a value of X'4040') or in 14-bit/16-bit mode (a value of X'0000').

To accommodate 14-bit/16-bit addressing, you must use NetSpy Level 4 software. As a temporary measure, you may modify the response time monitor user exit to force the address fields to 12-bit addressing until NetSpy Level 4 is installed.

Monitoring Other Applications with NetSpy

NetSpy can monitor any VTAM application while the user is in session with CL/SUPERSESSION. To monitor VTAM applications outside your NetSpy domain, add APPL statements to the NetSpy INITPRM member.

NetSpy will not recognize the session manager interface until either a user logs on or a session switch occurs (that is, until a virtual session becomes the active foreground session). If NetSpy is brought down and back up, the logical units (LUs) being displayed will be virtual terminals until the user switches sessions. Once the switch occurs CL/SUPERSESSION tells NetSpy the real LU name and that this LU is controlled by a session manager. You may need to use virtual session initialization dialogs with a short delay, so that NetSpy can detect the switch to foreground.

NetSpy does not support parallel sessions. Therefore, if you are implementing the NetSpy interface for the first time, you may need to enlarge your virtual terminal pools and change your application definitions. For more information, see “Virtual Terminals” on page 33 and “Virtual Terminal Pools” on page 41.

NetSpy must monitor the physical terminal in order to correlate its associated virtual sessions. Use the DT command to determine if the physical session is being monitored.

Parameters that may cause NetSpy not to monitor the session are:

- MAXLU
- MAXLU on the APPL statement
- TEXCLUDE
- MAXAPPL
- APPLNAME=NETNAME

Chapter 14. Storage

This section tells you how to monitor and reduce storage. The information in this section includes basic guidelines and general estimates. However, every configuration is unique, and some of our estimates and suggestions may not apply to your specific environment.

Several startup parameters control the allocation of storage for the CT/Engine address space:

- MAXIMUM
- MINIMUM
- RESERVE
- QUIESCE

For a detailed discussion of these parameters, see “CT/Engine Startup Parameters” on page 189.

Storage Allocation

The CT/Engine component of CL/SUPERSESSION is responsible for storage management. Storage is acquired from MVS at initialization, and then managed by the storage manager. This approach saves more CPU cycles than continual MVS GETMAIN and FREEMAIN.

At initalization, CT/Engine performs the following for both primary and extended storage.

1. If MINIMUM + RESERVE is greater than MAXIMUM, MAXIMUM is set to that total.
2. CT/Engine issues a conditional GETMAIN for the MAXIMUM value.
3. If the conditional GETMAIN succeeds, CT/Engine frees the storage requested and issues an unconditional GETMAIN for the value of MAXIMUM-RESERVE.

Monitoring Storage

To monitor storage, insert the following command in member KLSSTART of *&rhilev*.RLSCMDS:

```
EVERY 30:00 STORAGE
```

This logs a storage summary every 30 minutes.

Reducing Real Storage

You can reduce real storage by reducing virtual storage.

- Delete all unneeded virtual terminal pools.
- Turn off compression when possible. See “Compression” on page 231.
- LOGAPPL terminals to a gateway only when you cannot achieve the desired result in another way (for example, by using USS interpret tables). Inactive terminals that are LOGAPPLed to a gateway require virtual storage.
- When you use one-to-one virtual device mapping, as described in “Mapping Virtual Terminals” on page 52, issue VSM DEFINE commands dynamically (using the COMMAND dialog statement). You will save 0.5K primary storage for every unused virtual device, which would otherwise be defined and left in DEFER status. See the *Operator's Guide* for further discussion of the DEFER parameter.

  Also, if you know when a user exits a gateway and has no active sessions, issue a CLOSE command (using the COMMAND dialog statement) to close the virtual device and free all associated storage.

You can use storage isolation, also known as storage fencing, to reduce overall real storage requirements. A page rate of 8–15 pages per second in the CL address space is generally considered acceptable. For more information on storage isolation, see the IBM manual *Initialization and Tuning Guide*.

Storage requirements are mainly affected by terminal and session buffers, and less by the number of triggers and applications defined. Extended attribute byte (EAB) terminals require much more memory than do non-EAB terminals, because EAB buffers are twice as large. You can use administrator functions or dialog variables to control the allocation of EAB buffers. See the *Basic Configuration Guide* and the *Dialog Language Reference Manual*.

Extended Storage

Use extended storage if you can. See “CT/Engine Startup Parameters” on page 187 for descriptions of the MAXIMUM and MINIMUM startup parameters.

If it is available and specified, CL/SUPERSESSION and CL/GATEWAY use extended rather than primary storage for most storage requests.

Storage Preallocation

Member KLVINSTG of *&rhilev*.RLSPARM enables you to tune certain aspects of CL/SUPERSESSION storage management and reduce the product's overall requirement for real storage. KLVINSTG also allows preallocation of storage subpools in order to reduce working set size.

True working set size is the number of pages that must be in primary storage (central and extended) if CL/SUPERSESSION is to execute efficiently for a given workload. It is the amount of real storage the product requires in order to deliver good response time to a particular number of active users. Since true working set size is hard to measure, MVS reports working set as the number of pages that reside in primary storage at a given moment. This number may be larger or smaller than the true working set, but it can serve as a rough indication of true working set in a storage-constrained environment.

Storage managed by CT/Engine can become fragmented over time, leading to a larger working set than necessary. KLVINSTG provides a way to reduce this fragmentation by having CT/Engine preallocate an area of storage for small control blocks that are frequently obtained and freed.

KLVINSTG provides for preallocation of storage subpools with the default settings found in the KLVINSTG member of the initialization library *&rhilev*.RLSPARM.

The ACCESSED field of the output displays the number of times a block is allocated and released from a particular storage subpool. You can use this field to determine the relative activity of each storage subpool.

To preallocate storage subpools, use the following procedure.

1. Analyze the STORAGE DETAIL command output during peak periods of system activity. Look at the USE count for the small blocks, such as SIZE(1-16) and SIZE(17-32). The following is an example of STORAGE DETAIL output:

   ```
   KLVSD003    ALLOCATION DETAIL:
   KLVSD004       SIZE(1-16) USE(20) TOTAL(105) ACCESSED(1204)
   KLVSD004       SIZE(17-32) USE(0) TOTAL(1) ACCESSED(27)
   KLVSD004       SIZE(33-48) USE(5) TOTAL(6) ACCESSED(67)
   KLVSD004       SIZE(49-64) USE(1) TOTAL(2) ACCESSED(243)
   KLVSD004       SIZE(65-80) USE(1) TOTAL(8) ACCESSED(24696)

     ⋮
   KLVSD004       SIZE(16385-32768) USE(0) TOTAL(3) ACCESSED(101)
   ```

   The SIZE field shows the range (in bytes) of data block sizes in the subpool. For example, SIZE(1-16) indicates that this area contains all blocks that are from 1 to 16 bytes long. The USE field shows the number of blocks in use. The TOTAL field shows the total number of blocks that have been allocated.

   The important block sizes to focus on for preallocation are 16, 32, 64, 80, 256, and 512 byte blocks from extended storage (storage above the 16MB line).

   The KLVINSTG member contains the SIZE parameter with default settings that preallocate storage for block sizes 16 and 32. The format of the SIZE parameter is

   ```
   SIZE(nnnnnnn,mmmmmmm,c)
   ```

nnnnnnn	The number of bytes to preallocate per block (for example, 16 for SIZE 1-16).

mmmmmmm	The number of blocks to preallocate (the TOTAL value for that subpool block size).
c	The type of storage: primary (P) or extended (X).

Preallocating the smaller blocks (16- to 512-byte blocks) is critical to improving storage utilization, because these blocks tend to increase storage fragmentation. So, the smaller the block, the more benefits you may see with preallocation.

2. Use the default SIZE parameters in member KLVINSTG for your first run. The default settings of the SIZE parameter are

```
SIZE(16,100,P)
SIZE(16,500,X)
SIZE(32,500,X)
```

Note: When extended storage is available, you should not need to tune primary storage, since its use is minimal. If extended storage is not being used, follow standard STORAGE DETAIL analysis to establish parameter values.

3. During a period of peak activity, record the USE count for each block size (those listed in item 1 on page 215) returned by the extended portion of the STORAGE DETAIL command.

4. If the USE count for the small blocks is relatively high (a count over 1000), use this number to adjust the count of the default SIZE parameters in member KLVINSTG. You can enter SIZE parameters for each block that needs preallocation.

For example, the following display shows output from the extended STORAGE DETAIL command:

```
KLVSD002    EXTENDED MAIN STORAGE DETAIL
KLVSD003    ALLOCATION DETAIL:
KLVSD004       SIZE(1-16) USE(20) TOTAL(1629) ACCESSED(1204)
KLVSD004       SIZE(17-32) USE(0) TOTAL(1304) ACCESSED(27)
KLVSD004       SIZE(33-48) USE(5) TOTAL(32) ACCESSED(67)
KLVSD004       SIZE(49-64) USE(1) TOTAL(1191) ACCESSED(243)
KLVSD004       SIZE(65-80) USE(1) TOTAL(1545) ACCESSED(24696)

   ⋮
KLVSD004       SIZE(16385-32768) USE(0) TOTAL(3) ACCESSED(101)
```

With these calculations, specify the following storage configurations with the SIZE parameter in member KLVINSTG:

`SIZE(16,1625,X)`	Enter a count of 1625 for subpool 16; TOTAL count of 1629 rounded down to a multiple of 5.
`SIZE(32,1300,X)`	Enter a count of 1300 for subpool 32; TOTAL count of 1304 rounded down to a multiple of 5.
`SIZE(48,30,X)`	Enter a count of 30 for subpool 48; TOTAL count of 32 rounded down to a multiple of 5.

`SIZE(64,1190,X)`	Enter a count of 1190 for subpool 64; TOTAL count of 1191 rounded down to a multiple of 5.
`SIZE(80,1545,X)`	Enter a count of 1545 for subpool 80; TOTAL count of 1545 rounded down to a multiple of 5.

This example displays calculations for extended storage (X), but you would use the same procedure for primary storage (P).

If your working set size has been reduced, your preallocation has been successful. If not, you may want to run the system and try the calculations again.

It is important to define your preallocation sizes based on a loaded system. If you use numbers that are derived from off-hours, such as weekends, they will not be accurate for peak-hour usage when the counts are much higher.

If region storage defaults are changed (made smaller, for example), the preallocation default should be changed accordingly. If your environment changes—such as more users, or more sessions per user, or if you change the MINIMUM or MAXIMUM storage startup parameter in member KLSSYSIN—your preallocation should also be changed. In all cases, we recommend that you periodically re-evaluate your startup parameters.

Virtual Storage Requirements

When a table is opened or created, a structure called a *row set* is created in virtual storage. The row set contains the structure of the table and any rows contained within the table. Each table that is opened as SHARE has only one row set, regardless of the number of open requests against it. See “VTAM Options Initialization Parameters in KLVINVTM” on page 145 for more information.

The size of the row set in decimal bytes is calculated as follows:

Table 12. Virtual Storage Requirement Calculation (Row Set)

General overhead	160
Key and Name variable overhead	20/variable
Row overhead	36 + 8 /key or name + 12/extension
Row data	Actual length of key, name, and extension data
Sort overhead	24 + 20/field

Each OPEN or CREATE requires storage in addition to the row set size. This additional storage can be calculated as follows:

Table 13 (Page 1 of 2). Virtual Storage Requirement Calculation (OPEN or CREATE)

User-oriented blocks	80

Table 13 (Page 2 of 2). Virtual Storage Requirement Calculation (OPEN or CREATE)

TBSARG	4 + 20/condition
TBDISPL	24 + 4/row After row selection add 12/modified row + 14/modified variable + modified data length
TBGETX (locked row)	27/row
Temporary parameter block	112 + 24/variable name (if applicable)

VSAM Storage Requirements

When a table is written to the table database (TBCLOSE or TBSAVE), the required space in decimal bytes can be calculated as follows:

Table 14. VSAM Storage Requirement Calculation

General DASD overhead	142/VSAM record
Table overhead	314 + 17/key or name
Row overhead	20 + 12/key or name + 19/extension
Row data	Actual key, name, or extension data length
Sort overhead	10 + 10/field

The actual storage available in a VSAM record is the CI size (4096) less the overhead (142), or 3954. If a table requires more storage, additional VSAM records are written.

Note: These storage calculations do not include VSAM internal overhead, DASD blocking factors, FREESPACE, or any other non-table-related structures. For information, refer to the IBM manual *Managing VSAM Datasets*.

Chapter 15. Tables Unload/Load Facility

The tables unload/load facility unloads tables from the CT/Engine address space to PDS members and loads PDS members to CT/Engine tables. The facility allows tables to be moved from one CT/Engine application to another, for example, from a test system to a production system.

The unload/load facility provides

- A new SSPL dialog, KLVTBULD, to process unload/load requests.
- New line and pull-down commands in the CT/Engine operator facility to unload and load tables.

  Refer to the *Operator's Guide* for information on the operator facility.

Tables Unload/Load Utility Dialog

KLVTBULD, an SSPL dialog, unloads tables and loads PDS members.

KLVTBULD may be invoked from any other dialog, including non-terminal dialogs (NTDs). It performs *only* the actual load and unload processes and does not validate the appropriateness of a request. The calling dialog must determine whether a table or member should be overwritten.

KLVTBULD does not require a presentation space, and may be run as a non-terminal dialog (NTD).

Invoking KLVTBULD

KLVTBULD requires that a table and PDS name are passed to it; it returns information about the success or failure of the request.

When you unload a table, use the following dialog format:

```
dialog KLVTBULD (pack(0 'UNLOAD' '&tabhand' '&pds' 'userdlg' 'lrecl'))
unpack('&SysRC' rc reason msg)
```

When you load a table, use the following dialog format:

```
dialog KLVTBULD (pack(0 'LOAD' 'tabvar' '&pds' 'newname' 'userdlg'))
unpack('&SysRC' rc reason msg)
```

Input Parameters

UNLOAD	Unload the table to the PDS.
LOAD	Load the table from the PDS. The table must not exist on the tables database, nor be opened by any user.
&tabhand	The handle of the table to be unloaded to the PDS.
tabvar	The name of a variable to be updated with the handle of the table loaded from the PDS. If the table cannot be created, *tabvar* will be set to 0. *tabvar* must be declared SCOPE(SHARED).
&pds	The handle of the PDS member. For UNLOAD the PDS must be opened for WRITE (PDS SETWRT); for LOAD it must be opened for GET (PDS FIND). ***Note:*** The PDS data set must have an LRECL of 256 or higher, and a RECFM of V or VB. Any standard BLKSIZE is acceptable. By default, KLVTBULD segments the table data to fit a record size of 256 so that the data can be edited using ISPF. You can use a PDS with a larger LRECL to reduce DASD space overhead; specify the actual record size as *lrecl*.
newname	The desired table name. If null, the table name is taken from the unloaded data in *&pds*.
userdlg	The name of a dialog to be called at several points during the load or unload process. Refer to “User Dialog” on page 224 for details.
lrecl	The logical record length of *&pds*. Defaults to 256. If this value is greater than the actual record length, the unload will fail with a PDS WRITE error.

KLVTBULD returns a packed string in the *&sysrc* variable. It consists of three values:

rc	The KLVTBULD return code.
reason	The KLVTBULD reason code.
msg	The KLVTBULD message.

See “Return Codes” for explanations of each value.

Return Codes

0 Successful; the table has been loaded from, or unloaded to, *&pds*. *reason* is null. *msg* contains this information:

- The date the table was unloaded, as *yy/mm/dd*.
- The time the table was unloaded, as *hh:mm:ss*.
- User ID (*&vssuser*) requesting that the table be unloaded.
- Terminal ID (*&systerm*) where the unload request ran.

Each of these values is 8 characters long, separated by a single blank.

Note: The userid and/or terminal ID may be blanks. This occurs when the utility is run as a non-terminal dialog.

1	A table services request against the table being loaded or unloaded failed. *reason* is a text string that identifies the table operation, for example, TBGET. *message* contains the return code from the table services request.
2	A PDS services request against *&pds* failed. *reason* is a text string that identifies the PDS operation, for example, WRITE. *message* contains the return code from the PDS request.
3	Data structure error (LOAD only). *&pds* contains an invalid record. *reason* is the relative record number (1-based). *message* is the invalid record. The most likely error is that the member does not contain an unloaded table.
4	User dialog failed. *reason* indicates the type of failure: **1** Dialog does not exist or could not be refreshed. **2** Dialog failed during execution. *msg* is null.
5	The parameter list is in error. *reason* identifies the problem: **1** The first argument is not LOAD or UNLOAD. **2** *&tabhand* is not numeric. **3** *tabvar* is null or longer than 8 characters. **4** *newname* is longer than 44 characters. **5** *lrecl* is not numeric, is less than 256, or is more than 32767. *msg* is null.
6	The data in the PDS is missing information. *reason* identifies the problem: **1** End-of-file was unexpectedly reached. **2** The first record is not the control record. **3** A data value record was not preceded by a variable name record. *msg* is the PDS record number (1-based) of the record. **4** The number of rows in the control record does not match the number of rows in the PDS. **5** The number of key or name variables in the control record does not match the number of keys or names found in the PDS. **6** The data for a variable is longer than 32767. *msg* is the PDS record number (1-based) where the data would exceed the maximum.

Note: The actual maximum variable length depends on CT/Engine initialization values.

7 A key or name variable is invalid: either null or not separated from the following one by a space. *msg* is the record fragment being processed.

Except where indicated, *msg* is null.

7 The user dialog requested that the load or unload be terminated. *reason* is the value returned by *userdlg*. *msg* is the table row number (1-based).

Usage Notes

1. When you unload a table, sort information is not retained.
2. If your site is using a security package, such as RACF, the CT/Engine address space must have READ access to the PDS for LOAD and WRITE access for UNLOAD.
3. When KLVTBULD completes these conditions apply:
   - The table remains open. For a LOAD request, it has been opened WRITE and SHARE. Issue TBCLOSE or TBSAVE to save the table to the tables database.
   - The PDS is closed. When an UNLOAD request fails, the contents of the PDS member are unknown.
4. If KLVTBULD returns a TBCREATE failure, the *tabvar* variable is set to 0.
5. At the beginning of each load or unload request one or two messages are written to TLVLOG, identifying the table name, userid, and terminal. These messages create an audit trail and cannot be suppressed. See messages KLVDL201 and KLVDL202 in the *CL/GATEWAY for MVS and CL/GATEWAY for IMS Messages* for more information.
6. Although KLVTBULD will ensure that the data being loaded does not exceed 32767 in length, the actual maximum depends on the CT/Engine startup parameters and is generally around 30K.

UNLOAD Example

The following dialog is an example of how to code the UNLOAD process in a dialog.

```
set PDSName 'TEST.DATASET'              /* set names        */
set Member 'TSTTABLE'                   /*  for error msg   */

set PDSHand (pds('SETWRT'               /* open PDS WRITE   */
                 '&Member'              /* member           */
                 '*&PDSName'))          /* dsname           */
if &PDSHand < 0 do
```

```
  log('PDS SETWRITE for &PDSName(&Member) failed, RC=&PDSHand')
  return
end

set TabName 'MY.TABLE.NAME'               /* for error msg     */

set RC (tbopen('&TabName'                 /* open table        */
               1                          /* write=no          */
               1                          /* share=yes         */
               TabHand))                  /* put handle here   */
if &RC != 0 do
  log('TBOPEN for &TabName failed, RC=&RC')
  return
end

dialog KLVTBULD (pack(0 'UNLOAD'          /* unload request    */
                        &TabHand          /* from this table   */
                        &PDSHand))        /* to this PDS       */
unpack('&SysRC' RC                        /* get return code   */
                Reason                    /*  reason           */
                Message)                  /*   and message     */

if &RC = 0 set I 'successful'
else if &RC = 1 or &RC = 2 set I 'failed, &Reason request RC=&Message'
else set I 'failed, RC(&RC), REASON(&Reason), MESSAGE(&Message)'

log('UNLOAD for &TabName &I')

)term
if &TabHand tbend(&TabHand)               /* final cleanup     */
```

LOAD Example

The following is an example of how to code the LOAD process in a dialog.

```
set PDSName 'TEST.DATASET'                /* set names         */
set Member 'TSTTABLE'                     /*  for error msg    */

set PDSHand (pds('FIND'                   /* open PDS READ     */
                 '&Member'                /* member            */
                 '*&PDSName'))            /* dsname            */
if &PDSHand < 0 do
  log('PDS FIND for &PDSName(&Member) failed, RC=&PDSHand')
  return
end

dialog KLVTBULD (pack(0 'LOAD'            /* load request      */
                        TabHand           /* put handle here   */
                        &PDSHand          /* from this PDS     */
                        ''))              /* use name in pds   */

unpack('&SysRC' RC                        /* get return code   */
                Reason                    /*  reason           */
                Message)                  /*   and message     */

if &RC = 0 do
  tbsave(&TabHand)                        /* save to database  */
  tbname(&TabHand TabName)                /* get table's name  */
  set I 'successful, table name is &TabName'
end
else if &RC = 1 or &RC = 2 set I 'failed, &Reason request RC=&Message'
else set I 'failed, RC(&RC), REASON(&Reason), MESSAGE(&Message)'

log('LOAD &I')

)term
if &TabHand tbend(&TabHand)               /* final cleanup     */
```

User Dialog

A dialog may be specified to receive control with a DIALOG statement on both LOAD and UNLOAD requests. You can use this method to manage user data that is stored in the unloaded table. The dialog receives control at the following times:

- Before the first row is written to or read from the PDS
- After each row is written to the PDS or added to the table
- After the last row is written to the PDS (before the end-of-table record is written) or when the end-of-table record is read
- For LOAD, whenever a user record (type 8) is read from the PDS

The dialog is passed a packed string in *&sysparm* that contains the following information:

action A character string, LOAD or UNLOAD.

rownum Relative table row number (1-based).

-1	Indicates the beginning of the table.
-2	Indicates the end of the table.
-3	(LOAD only) Indicates a user record was read.

userdata For each value of *rownum* below:

-1	Contains the control record, not including the record type.
-2	Always null.
-3	(LOAD only) Contains the user record data, not including the record type.
nnn	Always null.

The user dialog must return to KLVTBULD with a RETURN with a packed string containing these values:

rc	One of the following:
	0 Continue the load or unload process.
	<>0 Terminate the load or unload. KLVTBULD terminates with *rc* set to 7 (*userdlg* requested termination), and *reason* set to this return code.
userdata	For an UNLOAD request, data to be written to the PDS as one or more user data records. If this record is longer than *lrecl*, it will be segmented as required; however, each segment will be returned independently on a subsequent LOAD request.
	This value is ignored for LOAD.

Sample User Dialog

The following example assumes that a shared variable, *&MyDesc*, contains a description of a table being processed.

This description is saved in a user record when the table is unloaded and is restored to the shared variable when the table is loaded.

```
)declare
 Action   scope(local)                      * request type
 RowNum   scope(local)                      * relative row #
 UserData scope(local)                      * data from record

 MyDesc   scope(shared)                     * table description

)init
 unpack('&SysParm' Action                   /* get action    */
                   RowNum                   /* row number    */
                   UserData))               /* and userdata  */

 if &RowNum = (neg 3) set MyDesc '&UserData'
 else if &RowNum = (neg 1) and
         &Action = 'UNLOAD' set UserData '&MyDesc'

 return (pack(0 0                           /* keep going    */
              '&UserData'))                 /* return data   */
```

Unloaded Data Structure

Each record in the PDS is variable length, up to the maximum length specified on the UNLOAD request (*lrecl*). The first byte indicates the type of record. The remainder depends on the type:

Type	Contents
0	Table control. Columns and their contents are as follows:

2	Data level. This is used to determine the structure of the unloaded data. The only valid value is 0.
3–46	Table name, left-justified, blank filled.
47–56	Number of rows, right-justified, zero filled.
57–66	Number of key variables, right-justified, zero filled.
67–76	Number of name variables, right-justified, zero filled.
77–84	Date the table was unloaded, as *yy/mm/dd*.
85–92	Time the table was unloaded, as *hh:mm:ss*.
93–100	Userid associated with unload request, left-justified, blank filled. The session variable *&vssuser* is used to obtain this value.
101–108	Terminal ID associated with unload request, left-justified, blank filled. The system variable *&systerm* is used to obtain this value.

1	Key variables. Each variable name is 8 characters long, left-justified, blank filled. The names begin in column 2 and are separated by a single blank. If there are too many name to fit in a record, another type 1 record immediately follows. If the table has no key variables, there will be no type 1 records.
2	Name variables. Each variable name is 8 characters long, left-justified, blank filled. The names begin in column 2 and are separated by a single blank. If there are too many names to fit in a record, another type 2 record immediately follows. If the table has no name variables, there will be no type 2 records.
3	Reserved for Candle future use. Ignored if present.
4	Variable name. Columns 2 through 9 contain the variable name, left-justified, blank filled. The variable may be key, name, or extension; the exact type may be determined by scanning the data in record types 1 and 2 for a match. If the name is not in those records, it is an extension variable.
5	Data value. Columns 2 through the maximum record length (*lrecl*) contain the data. If the data is less than the maximum, the record is appropriately shorter. If the data is longer than the maximum, additional type 5 records follow as needed. The end of the variable data is indicated by a record type other than 5.

6 End-of-row. There is no other data.

7 Reserved for Candle future use. Ignored if present.

8 User data. This record is created by request of the user dialog, *userdlg*. During a load request this data is passed to *userdlg*, if it was specified.

9 End-of-table. There is no other data. Any records following this are ignored.

All other record types cause KLVTBULD to quit processing and return a structural error code, rc=3.

Records must appear in the PDS in this order:

1. The first record must be table control, type 0.
2. The next record(s) must be key and/or name variables, types 1 and 2. These may occur in any order, and may be mixed. The number of key and name variables defined by record types 1 and 2 must match the numbers declared in the control record.
3. User records may appear at any point after the key- and name-variable records.
4. Each variable in a row consists of a type 4 record, which defines its name, and one or more type 5 records, which contain the variable's data.

   Type 4 and 5 records are not required when the variable is a key or name and the value is zero or a null. However, a KLVTBULD UNLOAD request will always generate type 4 and type 5 records for a null variable. The type 5 record will be one byte long and contain only the record type.

   A null extension variable requires a type 4 record for the name and a type 5 record that consists only of the record type.
5. At the end of each row is a type 6 record. The number of end-of-row records (type 6) must match the number of rows declared in the control record.
6. A type 9 record appears at the end of the table.

Chapter 16. Performance Tips

This chapter suggests ways to improve network performance and storage use with CL/GATEWAY and CL/SUPERSESSION. The information in this chapter includes basic guidelines and general estimates. However, every configuration is unique, and some of our estimates and suggestions may not apply to your specific situation.

Overall Performance Considerations

In general, the following considerations determine overall performance:

- Model of CPU installed

  For example, an Amdahl™ mainframe running in split-image mode will yield different performance figures from those of an IBM 3090-600E running in single-image mode.
- Levels of MVS (SP, XA, or ESA) and VTAM installed

  MVS/370 and versions of VTAM earlier than 3.1.0 preclude the use of extended storage and restrict the number of users.
- Number and type of terminals logged on
- Number and type of sessions being started
- Cumulative transaction rate of all active sessions
- Use of compression

Multiple-Processor CPUs

If the KLV region executes on a multiple-processor CPU (for example, a 3090™ model 200, which has two processors), specifying TASKS(1) in *&rhilev*.RLSPARM(KLSSYSIN) causes CT/Engine to use a single subtask, rather than one subtask per processor, for managing transaction throughput. This reduces overall CPU usage at the expense of throughput, since CT/Engine cannot exploit the other processor(s) or accomplish work on the same processor while waiting for paging.

Important

Use TASKS(1) only if paging is low.

By default, CT/Engine uses the same number of processors as the number of TASKS.

CPU Consumption

CPU consumption is mainly related to transaction rates; it is not affected by the number of users or by the number of sessions each user has.

CSA and ECSA Allocation

VTAM allocates CSA (or ECSA) on behalf of CT/Engine as it does for any other application. Each ACB requires approximately 1600 bytes of storage, each OPNDST another 400.

Here are some tips to ease VTAM allocation.

- Specify the minimum number of virtual terminals possible to VTAM. For recommendations, see the *Basic Configuration Guide*.
- Specify EAS=1 in the VTAMLST APPL statements, so that VTAM will not allocate a 2K hash table. The hash table is necessary only for virtual terminals that can have more than 30 simultaneous sessions (for example, virtual terminal pools whose VSM DEFINE command includes the PARALLEL parameter).

  For nonparallel pools, most virtual terminals will have fewer than 30 simultaneous sessions, and you can specify LIMIT=44 in the VSM DEFINE commands.

For more information, see

- “EAS Parameter” on page 37.
- “Virtual Terminal Sharing” on page 46.
- the *Operator's Guide*.
- IBM's *Network Program Products Storage Estimates*.
- IBM's *Virtual Storage Tuning Cookbook*.

APF Authorization

Authorize CT/Engine (and any associated TLVLOAD libraries specified in the CT/Engine startup procedure) so that CT/Engine becomes nonswappable and uses VTAM's authorized path.

SRB Exits

CT/Engine supports the VTAM authorized path and execution of VTAM exits in SRB mode, to reduce path length and overall CPU usage. If CT/Engine and the associated libraries are authorized, specify SRBEXIT=YES on all VTAMLST APPL statements associated with the KLV major node:

- CT/Engine
- all gateways
- all virtual device ACBs

SRBEXIT=YES specifies that some of the VTAM exit routines run in supervisor state, key zero (SRB mode), rather than in problem state (TCB mode).

Dispatching Priority

Make sure that CT/Engine has a dispatching priority immediately below VTAM and JES. No applications should have a dispatching priority higher than CT/Engine.

Application Status Monitoring

CL/GATEWAY and CL/SUPERSESSION let you specify the interval for monitoring application status. The parameter that controls the interval is MONITOR in *&rhilev.*RLSPARM(KLGINGWY). See “Setting CL/GATEWAY Initialization Parameters” on page 65.

If a large number of applications are defined to your system, you can lower CPU usage and network traffic by increasing the MONITOR interval.

Compression

CL/SUPERSESSION compression consists of the following:

- **Outbound Datastream Optimization**

  CL/SUPERSESSION removes repeated characters.

- **Screen Imaging**

  CL/SUPERSESSION maintains in memory an image of the physical terminal and sends changes only.

If you use compression only when necessary, you will save CPU cycles and eliminate one terminal buffer, which uses 2K to 8K of storage. (If available, extended storage is always used.)

The benefits of compression vary greatly with the application. ISPF already uses optimized datastreams; therefore, the benefit of compression is minimal.

Certain PC-based file transfer applications may work improperly with compression. These applications expect specific hex patterns in a specific order and cannot interpret a compressed 3270 datastream.

You can prevent file transfer problems by adding the \ft trigger, which calls dialog FXFER for the specified session and

- sets outbound data compression to No
- sets inbound data compression to No
- sets query PASSTHRU to Yes
- suspends timeout

The \ft trigger also inhibits immediate broadcast and sets

- read-modified for PA keys to No
- read-modified for ATTN keys to No
- fullread mode to N

The \ft trigger is a toggle; after file transfer is complete, the user can reissue \ft to reactivate data compression. For more information about adding the \ft trigger, refer to the *User's Guide*.

Printer data is never compressed.

The command

```
VSHOW userid STATS
```

shows the results of compression in real time. At virtual session termination, compression statistics are written to the Network Accounting Facility (NAF) dataset in the CL/SUPERSESSION virtual session termination record. See "CL/SUPERSESSION Record Types" on page 177.

Data Compression for Dialog Panels

Dialog panel displays are buffered much like the way virtual session screen images are buffered by CL/SUPERSESSION. This makes it possible for CT/Engine to provide automatic outbound data compression for panel displays, regardless of compression settings.

Adding Triggers

Candle recommends that you create no more than 40 triggers per user. This is the upper limit for trigger definitions before some noticeable amount of overhead may be created. The reason for the overhead is that an inbound message with the cursor located in one of the fields must be scanned for a possible trigger. Although the scanning process is efficient, a very long trigger list could require a noticeable amount of time to scan.

Appendix A. Logon Dialog Sequence

This appendix describes the dialog flow when a user logs onto CL/SUPERSESSION and CL/GATEWAY. The same sequence of dialogs takes effect when a user invokes the window control function to create a new window.

The KLGATEWY member, identified by the HOSTGATE commands in *&rhilev*.RLSCMDS(KLGCHGGW), is the first dialog in this sequence (shown in Figure 27). All dialogs shown in the figure are in *&rhilev*.RLSPNLS, the panel library.

| ***Note:*** The variable reference *n* indicates the national language identifier, which is added to the dialog name. For example, KLGLGON1 is the English language version of the logon dialog, KLGLGON*n*.

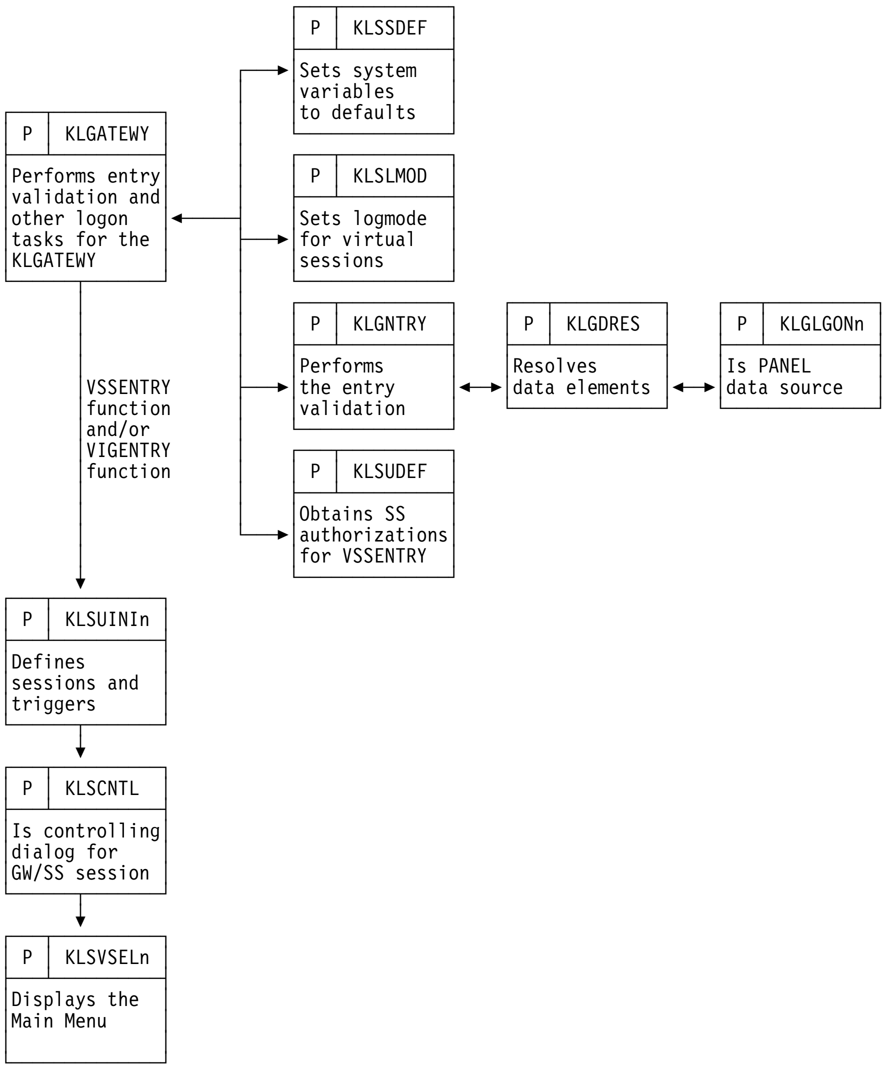
P KLSSDEF
Sets system variables to defaults
P KLGATEWY
Performs entry validation and other logon tasks for the KLGATEWY
P KLSLMOD
Sets logmode for virtual sessions
P KLGNTRY
Performs the entry validation
P KLGDRES
Resolves data elements
P KLGLGONn
Is PANEL data source
VSSENTRY function and/or VIGENTRY function
P KLSUDEF
Obtains SS authorizations for VSSENTRY
P KLSUINIn
Defines sessions and triggers
P KLSCNTL
Is controlling dialog for GW/SS session
P KLSVSELn
Displays the Main Menu

Figure 27. User Logon Process

KLGATEWY

This is the entry point dialog for CL/SUPERSESSION or CL/GATEWAY, executed when a user logs on.

KLSSDEF

Obtains the system defaults and authorizations necessary for entry validation.

KLSLMOD

During logon initialization, the value from the PSERVIC operand in the logmode table entry for the physical terminal determines session parameters such as screen size, SNA or non-SNA, and other device capabilities. This information allows virtual sessions to start with characteristics similar to those of the physical terminal session.

Use the administrator functions to customize your logmode table since the KLSLMOD dialog uses entries from that table. If you do not customize your logmode table, the dialog uses default logmodes.

Because you can use the administrator functions to customize your logmode table, you need never make changes to dialog KLSLMOD. For information on the administrator functions, see the *Basic Configuration Guide*.

KLGNTRY

Collects the USERID, PASSWORD, NEWPSWD, ACCOUNT, PROC, and GROUP data elements. Together with dialog KLGDRES, resolves these data elements according to the configuration defined in member KLGICFG*n* of *&rhilev*.RLSPARM.

KLGDRES

Together with dialog KLGNTRY, resolves data elements according to the configuration defined in member KLGICFG*n* of *&rhilev*.RLSPARM.

KLGLGONn

Displays a logon panel to collect the user ID, password, new password, group, account, and proc. KLGLGON*n* includes a one-digit national language identifier; for example, KLGLGON1 is the English language logon dialog.

KLSUDEF

Obtains defaults and authorizations necessary for entry validation.

KLSUINIn

Invoked when KLGATEWY executes the VIGENTRY function or the VSSENTRY function (CL/SUPERSESSION). Defines the initial sessions, triggers, and user profiles for the user. KLSUINI*n* includes a one-character national language identifier; for example, KLSUINI1 is the English language user initialization dialog.

KLSCNTL

This dialog receives control when a user logs on, or when a user leaves or terminates a foreground session. It is the controlling dialog that causes the Main Menu to be displayed.

KLSVSELn

Invoked by KLSCNTL. Displays the Main Menu. KLSVSEL*n* includes a one-digit national language identifier; KLSVSEL1 is the English language menu dialog.

Appendix B. Bringing Up a Test System

When you customize your system, you may want to test your modifications by bringing up a test copy of CL/SUPERSESSION on the same domain as the current production copy.

When you bring up a test system, many of the CL/SUPERSESSION and CT/Engine datasets can be shared. However, certain datasets and members must have unique names and cannot be shared.

The change requirements for a test system are listed below.

Table 15 (Page 1 of 2). Test System Requirements

Name	Required Change
VSAM Datasets	
NAM (*rvhilev*.RLSNAM)	Unique dataset name
TABLEDB (*rvhilev*.RLSTDB)	Unique dataset name
VIEWLOG (*rvhilev*.RLSVLOG)	Unique dataset name
Note: VIEWLOG is optional, but recommended by Candle.	
BSAM Datasets	
NAF (*rvhilev*.RLSNAF)	Unique dataset name
Initialization Datasets and Members	
rhilev.RLSPARM	Unique dataset name
KLVINNAF	DSNAME= must reference unique name
KLVINNAM	DSNAME= must reference unique name
KLVINTB	DSNAME= must reference unique name
KLVINVLG	DSNAME= must reference unique name
KLVINVPO	ACBNAME= must reference unique name
KLSLV002	Unique name; change in KLVINVPO
KLSSYSIN	Must reference unique name KLSSTART
Command Datasets and Members	
KLSSTART	Unique name; change in *rhilev*.RLSCMDS
KLSLV000	Unique name; change in KLSSTART
KLSLV001	Unique name; change in KLSSTART
KLSGW001	Unique name; change in KLGCHGGW
KLSGW002	Unique name; change in KLGCHGGW
KLSGW003	Unique name; change in KLGCHGGW

Table 15 (Page 2 of 2). Test System Requirements

Name	Required Change
KLS$VSMS	Unique name; change in *&rhilev*.RLSCMDS
SYS1.VTAMLST	Must contain unique names for virtual terminal definitions

Appendix C. CT/Engine, CL/SUPERSESSION, and CL/GATEWAY Custom Maintenance

Prior to Version 146, custom maintenance was shipped as ++USERMODs. These fixes were commonly known as USERMODs. In Version 147, custom maintenance will be shipped as ++APARs or ++PTFs, and will be known as custom fixes.

Table 16 lists all CT/Engine, CL/SUPERSESSION, and CL/GATEWAY custom maintenance for Version 115, Version 145, Version 146, and Version 147. If you are migrating from a version prior to Version 147, you can request any applicable custom fixes.

To identify the fixes you need for this release, find each fix or USERMOD that you received in a previous version, then look across the row to find the corresponding custom fix in the current version of the product. For example, if you are upgrading from Version 115 and have TA11008, you must request NLV1001 for Version 147.

When using this table, please consider the following.

- **N/A** means that this USERMOD is no longer applicable for this release level.
- **Base Product** means that the fix has been incorporated into that release of the product.
- When two or more USERMODs appear in one cell, the last USERMOD listed supersedes the previous USERMOD(s) for the same release level.

Table 16 (Page 1 of 4). Available USERMODs and Custom Fixes

Version 115	Version 145	Version 146	Version 147
TA11001	Base Product	Base Product	Base Product
TA11002, TA11020, TA11026, TA11043	TA14506, TA14535	QLV0143	QLV1063
TA11003, TA11036	TA14502	Base Product	Base Product
TA11004, TA11066	N/A	Base Product	Base Product
TA11005	Base Product	Base Product	Base Product
TA11006, TA11050	Base Product	Base Product	Base Product
TA11007	N/A	N/A	N/A
TA11008	N/A	TLV0005	NLV1001
TA11009	TA14500, TA14537, TA14543	QLV0195	QLV1053
TA11010, TA11051	Base Product	Base Product	Base Product
TA11011	Base Product	Base Product	Base Product
TA11012, TA11045, TA11064, TA11070	Base Product	Base Product	Base Product
TA11013	Base Product	Base Product	Base Product
TA11014, TA11023	Base Product	Base Product	Base Product
TA11015	Base Product	Base Product	Base Product
TA11016	TA14501, TA14526	Base Product	Base Product
TA11017	Base Product	Base Product	Base Product
TA11019, TA11034, TA11060, TA11065	N/A	QLV0222	QLV1093
TA11021	TA14503, TA14531, TA14546	Base Product	Base Product

Table 16 (Page 2 of 4). Available USERMODs and Custom Fixes

Version 115	Version 145	Version 146	Version 147
TA11022, TA11061	Base Product	Base Product	Base Product
TA11024	N/A	N/A	N/A
TA11025, TA11028, TA11029	Base Product	Base Product	Base Product
TA11027	N/A	TLV0021	Base Product
TA11030	Base Product	Base Product	Base Product
TA11031, TA11042, TA11054	N/A	TLV0059	NLV1002
TA11032, TA11052	Base Product	Base Product	Base Product
TA11033	TA14519	Base Product	Base Product
TA11035	Base Product	Base Product	Base Product
TA11037, TA11038, TA11044, TA11048, TA11076	TA14512	Base Product	Base Product
TA11039	Base Product	Base Product	Base Product
TA11040	Base Product	Base Product	Base Product
TA11041	Base Product	Base Product	Base Product
TA11046	Base Product	Base Product	Base Product
TA11049	Base Product	Base Product	Base Product
TA11053	N/A	N/A	N/A
TA11055	N/A	N/A	N/A
TA11056, TA11058	N/A	N/A	N/A
TA11057	N/A	TLV0026	NLV1003
TA11063, TA11067	N/A	N/A	N/A
TA11068, TA11072	Base Product	Base Product	Base Product
TA11069	N/A	N/A	N/A
TA11071	Base Product	Base Product	Base Product
TA11073	N/A	N/A	N/A
TA11074	N/A	N/A	N/A
TA11075	N/A	Base Product	Base Product
TA11077	N/A	Base Product	Base Product
TB11001, TB11008	Base Product	Base Product	Base Product
TB11002	Base Product	Base Product	Base Product
TB11003	Base Product	Base Product	Base Product
TB11004	N/A	N/A	N/A
TB11005	N/A	N/A	N/A
TB11006	N/A	N/A	N/A
TB11007, TB11009	N/A	N/A	N/A
TB11010	N/A	N/A	N/A
TB11011	N/A	N/A	N/A
TH11001	N/A	N/A	N/A
TH11002, TH11029, TH11047	Base Product	Base Product	Base Product
TH11003	N/A	N/A	N/A
TH11004	N/A	N/A	N/A
TH11005	N/A	N/A	N/A
TH11006	N/A	N/A	N/A
TH11007, TH11042, TH11051	Base Product	Base Product	Base Product
TH11008, TH11041	Base Product	Base Product	Base Product
TH11009	TH14500	TLV0015	NLU1001
TH11010, TH11034	Base Product	Base Product	Base Product
TH11011, TH11043, TH11058	N/A	N/A	N/A
TH11012	Base Product	Base Product	Base Product

Table 16 (Page 3 of 4). Available USERMODs and Custom Fixes

Version 115	Version 145	Version 146	Version 147
TH11013, TH11035, TH11039, TH11044, TH11061	N/A	N/A	N/A
TH11014, TH11040, TH11045, TH11059	Base Product	Base Product	Base Product
TH11015	N/A	N/A	N/A
TH11016	N/A	N/A	N/A
TH11017, TH11048	N/A	N/A	N/A
TH11018	N/A	N/A	N/A
TH11019, TH11036	N/A	N/A	N/A
TH11020, TH11025	Base Product	Base Product	Base Product
TH11021, TH11027	Base Product	Base Product	Base Product
TH11022, TH11063	Base Product	Base Product	Base Product
TH11023	N/A	N/A	N/A
TH11024	N/A	N/A	N/A
TH11026, TH11046	N/A	N/A	N/A
TH11028, TH11050, TH11056	TH14502, TH14510	TLV0017	NLU1002
TH11030	N/A	N/A	N/A
TH11031	N/A	N/A	N/A
TH11032	N/A	N/A	N/A
TH11033, TH11052, TH11060	N/A	N/A	N/A
TH11037	N/A	N/A	N/A
TH11038	N/A	N/A	N/A
TH11049	N/A	N/A	N/A
TH11053	N/A	N/A	N/A
TH11054	N/A	N/A	N/A
TH11055	N/A	N/A	N/A
TH11057	N/A	N/A	N/A
TH11062	N/A	N/A	N/A
TH11064	N/A	N/A	N/A
TH11065	N/A	N/A	N/A
N/A	TA14504	TLV0006	NLV1004
N/A	TA14505, TA14527, TA14547	TLV0001, TLV0054	NLV1005
N/A	TA14508	TLV0036	Base Product
N/A	TA14509	Base Product	Base Product
N/A	TA14510	Base Product	Base Product
N/A	TA14511, TA14544	TLV0053	NLV1006
N/A	TA14513	TLV0039	NLV1007
N/A	TA14514	Base Product	Base Product
N/A	TA14515, TA14542	TLV0004, TLV0011, TLV0052	Base Product
N/A	TA14516	Base Product	Base Product
N/A	TA14518, TA14530, TA14534, TA14538, TA14554	TLV0003, TLV0058	NLV1008
N/A	TA14521	TLV0042	NLV1009
N/A	TA14522	TLV0043	NLV1010
N/A	TA14525	TLV0044	N/A
N/A	TA14529, TA14549, TA14552	QLV0144	QLV1032
N/A	TA14532	N/A	N/A
N/A	TA14533	N/A	N/A
N/A	TA14536	N/A	N/A
N/A	TA14540	TLV0051	Base Product

Table 16 (Page 4 of 4). Available USERMODs and Custom Fixes

Version 115	Version 145	Version 146	Version 147
N/A	TA14548	QLV0195	QLV1053
N/A	TA14550	N/A	N/A
N/A	TA14551, TA14553	QLV0143	QLV1063
N/A	TB14500, TB14501	TLG0000	NLS1000
N/A	TD14500, TD14501, TD14502	TLI0000	NLI1000
N/A	TH14501, TH14519	TLV0016	NLV1011
N/A	TH14503, TH14511, TH14513, TH14518	N/A	NLU1003
N/A	TH14504, TH14507, TH14512	Base Product	Base Product
N/A	TH14505	Base Product	Base Product
N/A	TH14509	N/A	N/A
N/A	TH14514	No USERMOD	N/A
N/A	TH14515	No USERMOD	NLU1004
N/A	TH14516	No USERMOD	N/A
N/A	TH14517	No USERMOD	No USERMOD
N/A	TH14520	No USERMOD	No USERMOD
N/A	N/A	TLV0010	NLV1012
N/A	N/A	TLV0013	QLV1032

Appendix D. CL/ENGINE Product Libraries Reference

This appendix contains information about the CT/Engine product libraries that are user-modifiable.

TLVSYSIN

TLVSYSIN can contain one or more keywords that specify CT/Engine-wide processing values. It does not contain any product-specific information. Refer to the *Customization Guide* for a list of the valid TLVSYSIN keywords.

Syntax

Observe these syntax rules when coding keywords:

- TLVSYSIN can be either a fixed- or variable-length file.

  Note: All CT/Engine SMP/E-distributed maintenance assumes a fixed-length file with 80-byte records.

- Each physical record can contain a sequence number. Sequence numbers are detected according to these rules:

  Fixed-length files

  The last 8 characters of each record are examined.

  If both the first and last characters of this 8-character segment are numeric values, the record is assumed to contain a sequence number and those 8 columns are ignored. Otherwise, they are considered significant.

  Variable-length files

  The first 8 characters of each record are examined.

  If both the first and last characters of this 8-character segment are numeric values, the record is assumed to contain a sequence number and those 8 columns are ignored. Otherwise, they are considered significant.

- Complete keywords and any values specified for them on one physical record. Do not use continuations.
- You can specify more than one keyword on a physical record, separated by a comma or blanks. However, Candle does not recommend doing so, because complications for maintenance and debugging result.
- An asterisk in the first data position of a physical record causes the entire record to be treated as a comment and ignored.

- You can enter data in mixed case; it is always translated to upper case internally before processing.
- If you specify a keyword more than once, the last value generally is used. Refer to the description of each keyword for any different processing.
- You can specify keyword values with an equals sign or a set of parentheses. For example, the following keywords are equivalent:

  ```
  DEBUG=Y
  DEBUG(Y)
  ```

  Candle recommends that you use parentheses to delimit the data visually.

JCL Overrides

The values in TLVSYSIN are merged with two other sets of control values in this order:

1. Values defined internally. These values are used to set the basic CT/Engine defaults and are contained in the KLVST100 module.
2. Any values specified in the TLVSYSIN file.
3. Any values specified with the PARM= keyword of the JCL EXEC statement.

Because the *last* specified keyword value is the one that is used, you can use PARM= to specify keyword overrides without changing the TLVSYSIN dataset. The syntax rules are the same as for TLVSYSIN, with these exceptions:

- Separate keywords with either a comma or blanks.
- The PARM= string cannot be longer than 100 characters. This is an MVS restriction. (Refer to IBM's *JCL Reference* for more information.)
- MVS also requires that the PARM= string be enclosed in single quotes.
- Use of comments causes syntax errors.

Examples

Figure 28 on page 247 shows an example of a TLVSYSIN file.

```
************************************************************************
*                                                                      *
*  MEMBER: KLSSYSIN  CL/SUPERSESSION STARTUP PARAMETERS                *
*                                                                      *
*  FUNCTION:                                                           *
*                                                                      *
*     THIS MEMBER CONTAINS THE RECOMMENDED STARTUP PARAMETERS FOR THE  *
*     CL/SUPERSESSION ADDRESS SPACE.                                   *
*                                                                      *
************************************************************************
LSRPOOL(32768,3)
LSRPOOL(4096,32)
LSRPOOL(2048,8)
INITIAL(KLSSTART)
OPSTART(KLSOPST)
```

Figure 28. Typical TLVSYSIN File

An example of a JCL EXEC statement PARM= string follows.

```
//IEFPROC EXEC PGM=KLV,PARM='TRACE(12),DEBUG(Y)'
```

TLVPARM

TLVPARM contains configuration information for most CT/Engine components and for many CL/ENGINE-based products. Table 17 lists the CT/Engine-specific members.

Table 17. CL/ENGINE TLVPARM Members

Member	Component
KLUINVSS	Virtual Session Services
KLVINDM	Dialog Manager
KLVINNAF	Network Accounting Facility
KLVINNAM	Network Access Manager
KLVINPSM	Presentation Space Manager
KLVINRLM	Resource List Manager (invoked as part of NAM initialization)
KLVINSNS	Global Sense Code Processor
KLVINSTG	Storage Isolation
KLVINTB	Tables Manager
KLVINVLG	Viewlog
KLVINVPO	VTAM Program Operator
KLVINVSM	Virtual Session Manager
KLVINVTM	VTAM Support

Syntax

Observe these syntax rules when coding initialization members:

- TLVPARM can be either a fixed- or variable-length file. All libraries concatenated to the DD must have the same RECFM and LRECL.

  Note: All CT/Engine SMP/E-distributed maintenance assumes a fixed-length file with 80-byte records.

- Each CT/Engine or product component looks for only a specifically-named member. This is usually the same as the component's initialization module. (That is, the KLVINTB routine reads the KLVINTB member.)

- Each initialization member can contain sequence numbers. Sequence numbers are detected according to these rules:

  Fixed-length files
  The last 8 characters of the *first* record are examined.

  If all 8 characters are numeric, the *entire file* is assumed to contain sequence numbers and those 8 columns are ignored on *every* record. Otherwise, they are considered significant.

  Variable-length files
  The first 8 characters of the *first* record are examined.

  If all 8 characters are numeric, the *entire file* is assumed to contain sequence numbers and those 8 columns are ignored on *every* record. Otherwise, they are considered significant.

- Keywords and any values specified for them can span multiple physical records. Indicate a continuation by coding one of the following markers as the last nonblank character in a record:

  minus (-) The next line is concatenated to the current line with *one* blank separating the significant text. For example,

  ```
  KEYWORD1(VALUE1)    -
    KEYWORD2(VALUE2)
  ```

  is processed as

  ```
  KEYWORD1(VALUE1) KEYWORD2(VALUE2)
  ```

  plus (+) The next line is concatenated to the current line with all blanks retained. For example,

  ```
  KEYWORD1(VALUE1)    +
    KEYWORD2(VALUE2)
  ```

  is processed as

  ```
  KEYWORD1(VALUE1)      KEYWORD2(VALUE2)
  ```

Candle recommends that you use the minus/hyphen (-) because it slightly reduces parsing overhead. Use the plus sign (+) only when a specific number of blanks are required as, for example, in message text.

- You can specify more than one keyword on a physical record, separated by a comma or blanks. However, Candle does not recommend doing so, because complications for maintenance and debugging result.
- An asterisk that is not part of a quoted string causes the remainder of the *logical record* to be treated as a comment and ignored. For example,

  ```
  KEYWORD1(VALUE1)   * changed on 10/12/92 -
  KEYWORD2(VALUE2)
  ```

  is processed as

  ```
  KEYWORD1(VALUE1)   * changed on 10/12/92 KEYWORD2(VALUE2)
  ```

  and the KEYWORD2 keyword is ignored.
- You can enter data in mixed case; it is usually translated to upper case internally before processing. Refer to the description of each initialization member for any exceptions.
- Specify keywords once only, except where noted in the description of the initialization member.
- Specify keyword values either with an equals sign or a set of parentheses. For example, the following keywords are equivalent:

  ```
  TIMEOUT=10
  TIMEOUT(10)
  ```

  Candle recommends that you use parentheses to delimit the data visually.

Examples

Figure 29 shows an example of an initialization member in TLVPARM.

```
************************************************************************
*                                                                      *
*  MEMBER: KLVINTB   TABLE ACCESS MANAGER DATASET                      *
*                                                                      *
*  FUNCTION:                                                           *
*                                                                      *
*    THIS MEMBER IDENTIFIES THE VSAM DATASET USED BY THE TABLE         *
*    MANAGER TO STORE TABLES.                                          *
*                                                                      *
************************************************************************
&rvhilev.RLSTDB -
DISP(SHR)
```

Figure 29. Typical TLVPARM Member

TLVCMDS

TLVCMDS contains command lists (CLISTs) that may contain one or more CT/Engine or product operator commands and/or invocations of other CLISTs. CT/Engine executes each logical record in a CLIST as if it had been entered at an operator's console. Thus, a set of related commands, such as those issued during CT/Engine initialization, can be grouped together and invoked by just one command.

Refer to the *Operator's Guide* for information about CT/Engine operator commands.

Syntax

Observe these syntax rules when coding CLISTs:

- TLVCMDS can be either a fixed- or variable-length file. All libraries concatenated to the DD must have the same RECFM and LRECL.

  Note: All CT/Engine SMP/E-distributed maintenance assumes a fixed-length file with 80-byte records.

- CLIST member names are 1 to 8 characters long. For compatibility with previous CT/Engine releases, they may begin with a dollar sign ($). Do not give a CLIST the same name as a CT/Engine or product operator command. Operator commands are always searched for first.
- Invoke CLISTs by entering the CLIST name *without* a preceding dollar sign.
- Each CLIST member can contain sequence numbers. Sequence numbers are detected according to these rules:

  Fixed-length files
  The last 8 characters of the *first* record are examined.

  If all 8 characters are numeric, the *entire file* is assumed to contain sequence numbers and those 8 columns are ignored on *every* record. Otherwise, they are considered significant.

  Variable-length files
  The first 8 characters of the *first* record are examined.

  If all 8 characters are numeric, the *entire file* is assumed to contain sequence numbers and those 8 columns are ignored on *every* record. Otherwise, they are considered significant.

- Commands and their keywords may span multiple physical records. Indicate a continuation by coding one of these markers as the last nonblank character in a record:

  minus (-) The next line is concatenated to the current line with *one* blank separating the significant text. For example,

```
CMD OP1  -
  KEYWORD2(VALUE2)
```

is processed as

```
CMD OP1 KEYWORD2(VALUE2)
```

plus (+) The next line is concatenated to the current line with all blanks retained. For example,

```
MYCMD OP1  +
  KEYWORD2(VALUE2)
```

is processed as

```
MYCMD OP1    KEYWORD2(VALUE2)
```

Candle recommends that you use the minus/hyphen (-) because it slightly reduces parsing overhead. Use the plus sign (+) only when a specific number of blanks are required as, for example, in message text.

- Specify only one command or CLIST invocation on a logical record.
- An asterisk that is not part of a quoted string causes the remainder of the *logical record* to be treated as a comment and ignored. For example,

```
CMD OP1            * changed on 10/12/92 -
KEYWORD2(VALUE2)
```

is processed as

```
CMD OP1            * changed on 10/12/92 - KEYWORD2(VALUE2)
```

and the KEYWORD2 keyword is ignored.

- The PROFILE operator command determines how commands that you enter are treated. If `PROFILE FOLD` has been specified, all data is translated to upper case; otherwise the data is left as it was entered. This may cause problems for some commands that do not fold their operands.

  Note: The operator logon CLIST, *&thilev.*TLSCMDS(KLSOPST), issues `PROFILE FOLD`.

- Specify command keywords once only, except where noted in the description of the command.
- Specify keyword values either with an equals sign or a set of parentheses. For example, the following keywords are equivalent:

```
POOL=TSO
POOL(TSO)
```

Candle recommends that you use parentheses to delimit the data visually.

Examples

Figure 30 shows an example of a CLIST member in TLVCMDS.

```
************************************************************************
*                                                                      *
*  MEMBER: KLSSTART  CL/SUPERSESSION STARTUP                           *
*                                                                      *
*  FUNCTION:                                                           *
*                                                                      *
*    THIS COMMAND LIST INITIALIZES CL/SUPERSESSION.                    *
*                                                                      *
************************************************************************
TRACE +DISPATCH          * ACTIVATE DISPATCHER TRACE
EVERY 2:00               * PREVENT S522 ABENDS
EVERY 30:00 STORAGE D    * LOG STORAGE USE EVERY 30 MINUTES
EVERY 30:00 FLUSH        * FLUSH VSAM LSR BUFFERS EVERY 30 MINUTES
NODE &LSVT0              * START CL/ENGINE STANDARD OPERATOR FACILITY
DIALOG &LSVT5 KLVLOGON   * START CL/ENGINE CUA OPERATOR FACILITY
VSM DEF LU1 &LSVT0 TH(9) LOGMODE(SCS) DEFER
```

Figure 30. Typical TLVCMDS Member

TLVPNLS

TLVPNLS contains SSPL dialogs used by CT/Engine and CL/SUPERSESSION. Refer to the following documents for information about coding SSPL dialogs.

- *SSPL Programming Guide*
- *Dialog Language Reference Manual*

TLVLOAD

TLVLOAD contains CT/Engine and CL/SUPERSESSION load modules. It may also contain user exit routines if you have implemented them. Table 18 lists the CT/Engine-specific modules that you can implement, and where they are documented.

Table 18 (Page 1 of 2). CT/Engine TLVLOAD Load Modules

Module	Description	Reference
KLSA2INH	Used with the LINK dialog function to implement ACF2 logon inheritance.	Customization Guide
KLSA2NEV	NAM exit routine for ACF2 (MVS only).	Customization Guide

Table 18 (Page 2 of 2). CT/Engine TLVLOAD Load Modules

Module	Description	Reference
KLSNAFPT	NAF print routine. (This module does not have to be in TLVLOAD.)	Customization Guide
KLSNAF15	NAF print routine sort exit. (This module does not have to be in TLVLOAD.)	Customization Guide
KLSNAMPT	NAM print routine. (This module does not have to be in TLVLOAD.)	Customization Guide
KLSNAMPX	NAM FIELDEXIT routine, to validate the syntax of security data.	Customization Guide
KLSNAMX	A basic NAM validation exit.	Customization Guide
KLSTSNEV	NAM exit routine for CA-TOP SECRET.	Basic Configuration Guide
KLSUSR20	An IPCS/AMDPRDMP verb exit for formatting CT/Engine trace records. (This module does not have to be in TLVLOAD.)	Problem Determination Guide
KLSXNPM	NPM interface exit, to determine the account code to be passed to NPM.	Customization Guide
KLSXRTM	RTM interface exit, to determine the mapping message that is sent to the physical terminal.	Customization Guide
username	A user module, invocable with the LINK dialog function.	Dialog Language Reference Manual

Syntax

Each of the modules listed in the above table must be written in IBM assembler language. Refer to the indicated document for information about coding, modifying, and implementing the exits.

Index