

AllFusion™ Endevor® Change Manager

Interface for CA-Netman®
Administration Guide
4.0



Computer Associates®

SP1
ENN400

This documentation and related computer software program (hereinafter referred to as the "Documentation") is for the end user's informational purposes only and is subject to change or withdrawal by Computer Associates International, Inc. ("CA") at any time.

This documentation may not be copied, transferred, reproduced, disclosed or duplicated, in whole or in part, without the prior written consent of CA. This documentation is proprietary information of CA and protected by the copyright laws of the United States and international treaties.

Notwithstanding the foregoing, licensed users may print a reasonable number of copies of this documentation for their own internal use, provided that all CA copyright notices and legends are affixed to each reproduced copy. Only authorized employees, consultants, or agents of the user who are bound by the confidentiality provisions of the license for the software are permitted to have access to such copies.

This right to print copies is limited to the period during which the license for the product remains in full force and effect. Should the license terminate for any reason, it shall be the user's responsibility to return to CA the reproduced copies or to certify to CA that same have been destroyed.

To the extent permitted by applicable law, CA provides this documentation "as is" without warranty of any kind, including without limitation, any implied warranties of merchantability, fitness for a particular purpose or noninfringement. In no event will CA be liable to the end user or any third party for any loss or damage, direct or indirect, from the use of this documentation, including without limitation, lost profits, business interruption, goodwill, or lost data, even if CA is expressly advised of such loss or damage.

The use of any product referenced in this documentation and this documentation is governed by the end user's applicable license agreement.

The manufacturer of this documentation is Computer Associates International, Inc.

Provided with "Restricted Rights" as set forth in 48 C.F.R. Section 12.212, 48 C.F.R. Sections 52.227-19(c)(1) and (2) or DFARS Section 252.227-7013(c)(1)(ii) or applicable successor provisions.

First Edition, June 2003

© 2003 Computer Associates International, Inc.
All rights reserved.

All trademarks, trade names, service marks, and logos referenced herein belong to their respective companies.

Contents

Chapter 1. Interface Overview	1-1
1.1 Overview	1-2
1.2 Resources Needed to Implement the Endevor Interface to CA-Netman	1-3
1.3 Interface Functionality	1-4
1.4 Interface Operations	1-5
1.4.1 Action Processing	1-7
1.4.2 Package Processing	1-7
Chapter 2. Chapter 2: Activating the Interface	2-1
2.1 Overview	2-2
2.2 Endevor Defaults Table - C1DEFLTS	2-3
2.3 Add Netman Executable to Endevor CLIST	2-4
2.4 BC1TNM90 Definition Table	2-5
2.5 BC1PNM60 Log Program	2-8
2.6 Trace Facility	2-10
2.6.1 EN\$TRNMI Trace	2-10
Chapter 3. Setting Up the CA-Netman Interface	3-1
3.1 Overview	3-2
3.2 Defining Endevor CCIDs to CA-Netman	3-3
3.2.1 CA-Netman Change Record	3-3
3.2.1.1 Endevor Change Online View Example	3-4
3.2.2 CA-Netman Task Records	3-4
3.2.2.1 CA-Netman Task Log Records	3-5
3.2.2.2 Endevor Task Online View Example	3-8
3.2.2.3 Endevor Task Log Online View Example	3-9
3.3 Defining Endevor Packages to CA-Netman	3-10
3.3.1 CA-Netman Change Record	3-10
3.3.1.1 CA-Netman Change Log Record	3-11
3.3.1.2 Endevor Change Online View Example	3-15
3.3.1.3 Endevor Change Log Online View Example	3-16
Chapter 4. Verifying Endevor Activities Against CA-Netman	4-1
4.1 Overview	4-2
4.2 Element Action Processing	4-3
4.2.1 Before Element Action	4-3
4.3 Package Processing	4-4
4.3.1 Before Package Function	4-4

Chapter 5. Logging Endevor Activities in CA-Netman	5-1
5.1 Overview	5-2
5.2 After Element Action	5-3
5.3 After Package Function	5-4
5.4 Modifying Log Information	5-5
5.4.1 CA-Netman Record Population	5-5
Appendix A. BC1PNM60 Source Module	A-1
A.1 BC1PNM60	A-2

Chapter 1. Interface Overview

1.1 Overview

This chapter discusses the functionality and operations of the AllFusion™ Endevor® Change Manager Interface for CA-Netman® (referred to from hereon as the Endevor Interface to CA-Netman, CA-Netman or the Endevor CA-Netman Interface).

1.2 Resources Needed to Implement the Endevor Interface to CA-Netman

Implementing the Endevor interface to CA-Netman requires that you make decisions about which Endevor information you want to track in CA-Netman, and where you want to store that information in CA-Netman. Therefore, to implement the interface most effectively, you should have the following resources available:

- People familiar with how Endevor is used at your site, and in particular with Endevor packages and exits
- People familiar with CA-Netman policies and operations at your site

Information is available from the following publications:

- Unicenter® CA-Netman *System Programmer Guide*
- Unicenter CA-Netman *Techniques Guide*
- Unicenter CA-Netman *Reference Guide - Part 1*
- AllFusion Endevor Change Manager *Exits Guide*
- AllFusion Endevor Change Manager *Packages Guide*

Note: See your CA-Netman administrator for questions about site-specific information.

1.3 Interface Functionality

The Endevor CA-Netman Interface provides a mechanism for controlling Endevor element actions and package functions based on CA-Netman's database record structures. It also provides a mechanism for logging Endevor activities in CA-Netman.

In addition to managing the initialization and termination of CA-Netman sessions under Endevor, the interface implements package-level and action-level interaction between Endevor and CA-Netman. This allows:

- The use of CA-Netman as an auditing tool. Through the Interface Definition Table, you can control the levels of logging (that is, to log an element action or package function activities or both).
- Control over CA-Netman data record population. The source module that actually updates CA-Netman records will be distributed. This gives you some flexibility on how data is populated onto CA-Netman records.
- Control Action processing by validating element action CCID's through CA-Netman.
- Control Package processing by predefining package IDs through CA-Netman.

Note: Any Endevor field or text string that uses the underscore (_) character appears as a blank in Netman. The underscore character is valid for use in an Endevor HFS element name.

1.4 Interface Operations

The CA-Netman Interface is implemented through Endevor's exits and is centered around CA-Netman's change, task, and log records. You activate the interface by specifying 'Y' in the NMAN parameter in the C1DEFLTS table.

Note: Refer to Chapter 2, "Activating the Interface," for information about the C1DEFLTS table.

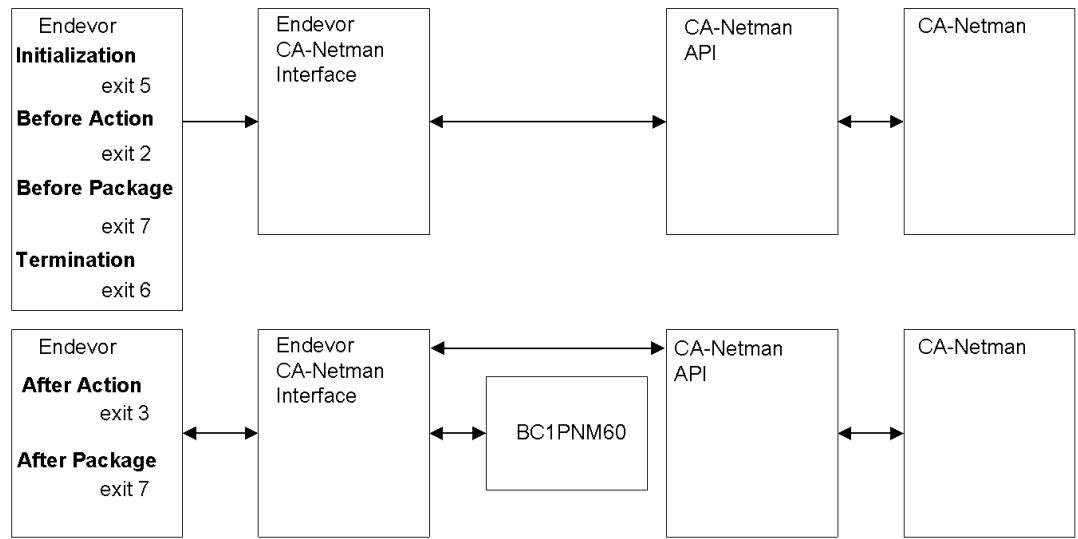
Endevor provides a full set of exit points (before exit and after exit) for actions and packages and uses control blocks to provide information to user programs written for these exit points.

Exit Point	Name
2	Before Action
3	After Action
7	Before Packages After Packages

Note: Refer to the AllFusion Endevor Change Manager *Exits Guide* information about exits.

The CA-Netman Interface communicates directly from Endevor exit points with the CA-Netman API. The CA-Netman API module NTMAPI establishes a CCI connection with CA-Netman. Information and command strings flow across this connection allowing Endevor access to the CA-Netman database. This architecture allows Endevor to access the CA-Netman database which might be running on any computer in the network.

The following diagram summarizes the interface operation.



During exit initialization, the CA-Netman Startup Definition table is loaded. This table is referenced whenever the interface invokes the CA-Netman API. After the load, the INFOMAN and NETMAN default options are examined. (If both options are turned on, non-display package and element action processing follow the behavior based on the startup table's "Netman Required" indicator (see paragraph below). After this check, the interface attempts a Signon API request.)

For errors, with the "Netman Required" indicator turned on, the interface does NOT allow any element actions (except print and list) and any non-display package functions to execute. If the "Netman Required" indicator is turned off, Endevor processing continues normally but all interface activity is suspended.

During exit 3 and exit 7 (after actions and package functions), log, change, and task records may be created when the startup table's log indicators are on. While populating CA-Netman records, a common module, BC1PNM60, is called to do the actual data population in CA-Netman's API request buffers. This module is distributed in source so that you may tailor (change) record population to meet the needs of your site.

During exit termination, the interface invokes a Signoff API request. At this time, all storage, tables, etc. are freed.

1.4.1 Action Processing

The interface implements action processing by validating element action CCIDs through CA-Netman.

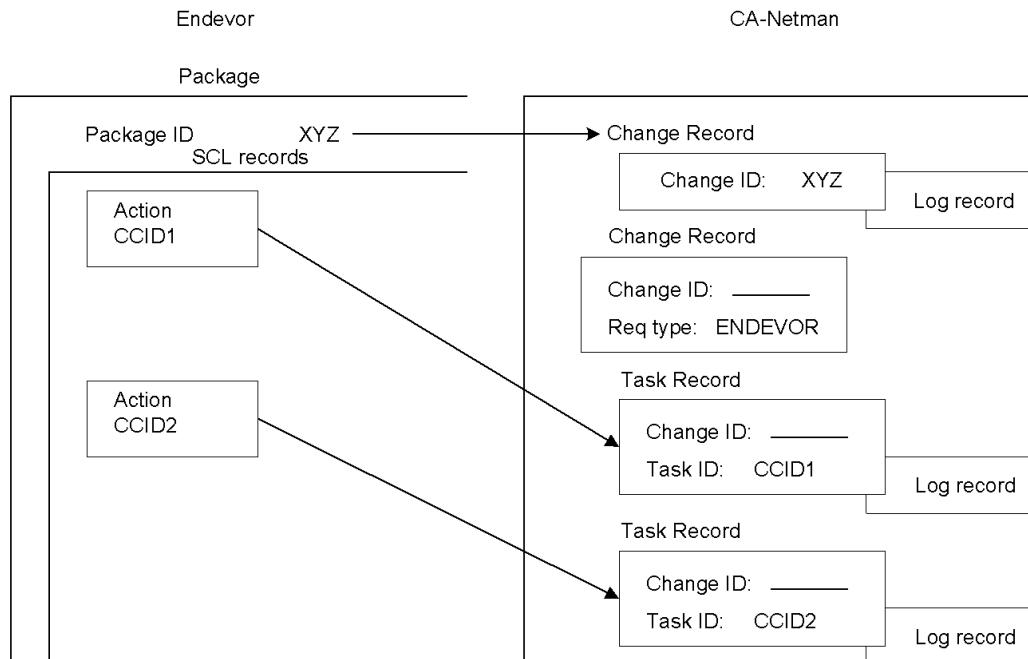
Note: Refer to Chapter 4, "Verifying Endevor Activities Against CA-Netman," for more information about action processing.

1.4.2 Package Processing

The interface implements package processing by predefining package IDs through CA-Netman.

Note: Refer to Chapter 4, "Verifying Endevor Activities Against CA-Netman," for more information about package processing.

The diagram below summarizes the relationship between Endevor activities (both validation and tracking) and CA-Netman records used to control Endevor functions.



Chapter 2. Chapter 2: Activating the Interface

2.1 Overview

This chapter discusses how to activate the CA-Netman Interface and customize your site by using the Endevor Defaults Table, C1DEFLTS, adding Netman data sets to CLIST that will bring up Endevor, the BC1TNM90 definition table, the BC1PNM60 log program, and the CA-Netman Interface trace facility.

2.2 Endevor Defaults Table - C1DEFLTS

You can activate CA-Netman by typing **Y** in the NMAN parameter value in the Endevor C1DEFLTS table.

The Netman database must be up and CCI needs to be active in order to activate CA-Netman.

Note: The CA-Netman interface can NOT be active along with the Info/Man interface. If both parameters are specified, Endevor will fail to initialize CA-Netman. Also, if both are specified, the C1DEFLTS table will NOT assemble.

A section of the C1DEFLTS table is shown below. A 'Y' value specified on the NMAN parameter will activate the interface.

C1DEFLTS TYPE=MAIN,		
ACCSTBL=,	ACCESS SECURITY TABLE	X
APRVFLG=N,	APPROVAL PROCESSING (Y/N)	X
ASCM=N,	ASCM CONTROL ACTIVATION	X
BATCHID=0,	BATCH UID FROM JOBNAME/USER=	X
CIPODSN=,	CCID VAILIDATION DSN	X
CSP=N,	CSP CONTROL ACTIVATION	X
CUNAME='*** PUT YOUR COMPANY NAME HERE ***',		X
DB2=N,	DB2 CONTROL ACTIVATION	X
ELINK=N,	ENDEVOR/LINK CONTROL ACTIVATION	X
ESSI=N,	ESSI CONTROL ACTIVATION	X
INFO=N,	INFOMAN CONTROL ACTIVATION	X
NMAN=Y,	NETMAN CONTROL ACTIVATION	X
LIBENV=,	LIBRARIAN (LB), PANVALET (PV)	X
LIBENVP=N,	LIB/PAN CONTROL ACTIVATION	X
LIBRPGM=,	LIBRARIAN BATCH PROGRAM NAME	X
LINESPP=60,	LINES PER PAGE	X
MACDSN='iprfx.iqual.SOURCE',	E/MVS SOURCE LIBRARY	X
PKGDSN='uprfx.uqual.PACKAGE',	PACKAGE DATASET NAME	X
PKGTSO=Y,	FOREGROUND PACKAGE EXEC (Y/N)	X

2.3 Add Netman Executable to Endevor CLIST

You need to add the Netman loadlib to the CLIST you use to bring up Endevor.

Note: This data set should be concatenated as an AUTHLIB.

2.4 BC1TNM90 Definition Table

You must create the BC1TNM90 start-up definition table. To do this, you must edit source module BC1TNM90 and update the @ENETM parameters. Then you are required to assemble and link it into a library that is accessible to Endevor.

The @ENETM macro is shown below.

```
@ENETM START,
  SIGNONID=XXX,
  PASSWORD=XXXXXXXX,
  DBNAME=XXXXXXXXXXXX,
  WAITTIME=300,
  OPERLOGD=XX,
  VALCCID=Y,
  LOGACT=Y,
  VALPKG=Y,
  LOGPKG=Y,
  CATEGORY=XXXXXXXX,
  NMNRQD=Y
@ENETM END
```

The entries in this macro are described below.

Parameter	Description
ENTRY=START	Indicates the beginning of the BC1TNM90 table.
SIGNONID=	The CA-Netman API requires a signon. A signon ID must be specified. This value can be up to three characters long. There is no default value.
PASSWORD=	The CA-Netman signon password. A password must be specified. This value can be up to eight characters long. There is no default value.
DBNAME=	The CA-Netman database name. The name must be specified. The AK0 prefix should not be included in the name specification. This value can be up to thirteen characters long. There is no default.
WAITTIME=	CA-Netman wait time in seconds. This parameter defines how long Endevor will wait for a CA-Netman API request to complete. This is a four-character field and has a maximum specification value of 3539 seconds. The default value is 300 seconds.
OPERLOGID=	Required. CA-Netman's operational log ID. Used when creating log records for change and task records. This is a two-character field. There is no default.
VALCCID=	CCID validation flag. A one character flag. May be set to 'Y', for yes, or 'N', for no. The default value is 'Y'.

Parameter	Description
LOGACT=	Log action activity flag. A one-character flag. May be set to 'Y,' for yes, or 'N,' for no. The default value is 'Y.' When set to Y, activity log records will be created and attached to their corresponding Task record.
VALPKG=	Package validation flag. A one-character flag. May be set to 'Y,' for yes, or 'N,' for no. The default value is 'Y.'
LOGPKG=	Log package activity flag. A one-character flag. May be set to 'Y,' for yes, or 'N,' for no. The default value is 'Y.' When set to Y, activity log records will be created and attached to the proper package change record.
CATEGORY=	CA-Netman's Category field. This value will be used when Endevor is required to create a change or task record. If not specified, all CCID task records and Package change records must be created prior to any Endevor element action and Package function activities. Note: This field is site-specific in CA-Netman.
NMANREQD=	CA-Netman required indicator. This indicator tells Endevor how to behave when CA-Netman error's occur. If set to 'N,' Endevor will continue to run normally but will make no further attempts to communicate to CA-Netman. CCID validation will only be done on the Endevor side and Package validation will not be done. If set to 'Y,' all Endevor activities will be canceled once CA-Netman error's occur. This is a one-character field and may be set to 'Y,' for yes, or 'N,' for no. The default value ID 'Y.'
ENTRY=END	Required. This statement denotes the end of the BC1TNM90 table.

Note: A copy of the BC1TNM90 source module can be found in the Endevor library iprfx.iqual.SOURCE. BC1JNM90 sample JCL can be found in the Endevor data set iprfx.iqual.JCLLIB.

The sample JCL for BC1JNM90 follows:

```

///* ( COPY JOBCARD )
//***** *****
//*
//*      BC1JNM90 - THIS JOB IS USED TO CUSTOMIZE AND BUILD THE      *
//*                  ENDEVOR-OS/390 NETMAN INTERFACE USER DEFINITION      *
//*                  TABLE CALLED BC1TNM90.                                *
//*
//*      THE FOLLOWING UPDATES MUST BE MADE TO THIS JCL BEFORE      *
//*      IT CAN BE EXECUTED:                                         *
//*
//*      1. UPDATE THE JOBCARD TO REFLECT CORRECT SITE INFORMATION    *
//*      2. REVIEW THAT THE SOURCE AND LOADLIB DATA SET NAMES ARE      *
//*          CORRECT.                                                 *
//*          - iprfix.iqual.SOURCE                                     *
//*          - uprfx.uqual.LOADLIB                                     *
//*      3. TAILOR YOUR SITE'S ENDEVOR/NETMAN PARAMETERS BY EDITTING   *
//*          MEMBER BC1TNM90 IN iprfix.iqual.SOURCE.                   *
//*
//***** *****
//STEP1    EXEC PGM=ASMA90,
// PARM='NODECK,OBJECT,NOTERM,LIST,XREF(SHORT),NOUSING,LANGUAGE(UE)'
//SYSLIB   DD  DISP=SHR,DSN=iprfix.iqual.SOURCE
//          DD  DISP=SHR,DSN=SYS1.MACLIB
//SYSLIN   DD  DISP=(NEW,PASS,DELETE),DSN=&&SYSLIN,
//          UNIT=tdisk,SPACE=(TRK,(3,5),RLSE),
//          DCB=(RECFM=FB,LRECL=80,BLKSIZE=3120)
//SYSPUNCH DD  DUMMY
//SYSUT1   DD  UNIT=tdisk,SPACE=(CYL,(5,3))
//SYSPRINT DD  SYSOUT=*
//SYSIN    DD  DISP=SHR,DSN=iprfix.iqual.SOURCE(BC1TNM90)
//***** *****
//* BC1TNM90 SHOULD BE LINKED INTO YOUR SITE'S LOADLIB DATASET. THIS *
//* LIBRARY MUST BE ACCESSIBLE TO ENDEVOR.                               *
//***** *****
//*
//STEP2    EXEC PGM=IEWL,PARM='LIST,NCAL,RENT,XREF,SIZE=(256K,64K)',
//          COND=(0,NE)
//SYSLIN   DD  DISP=(OLD,DELETE),DSN=&&SYSLIN
//SYSLMOD  DD  DISP=SHR,DSN=uprfx.uqual.LOADLIB(BC1TNM90)
//SYSUT1   DD  UNIT=tdisk,SPACE=(CYL,(5,3))
//SYSPRINT DD  SYSOUT=*

```

2.5 BC1PNM60 Log Program

During the CA-Netman record population process for change, task, and log records, module BC1PNM60 is called to build the CA-Netman API command strings to populate these record fields. This module is included in source so you may alter record population of these records to tailor your site's needs.

Note: Refer to Chapter 5, "Logging Endevor Activities in CA-Netman," for information about customizing BC1PNM60. Also see the AllFusion Endevor Change Manager *Exits Guide* to understand which Endevor control blocks (information) are available to you during action and package processing.

The following sample JCL, BC1JNM60, may be used to assemble and link source module BC1PNM60.

Note: A copy of the BC1PNM60 source module can be found in the Endevor library iprfx.iqual.SOURCE. BC1JNM60 sample JCL can be found in the Endevor data set iprfx.iqual.JCLLIB.

```
/* ( COPY JOBCARD )
*****
/*      BC1JNM60 - THIS JOB IS USED TO ASSEMBLE AND LINK THE      *
/*          ENDEAVOR-OS/390 NETMAN INTERFACE POPULATION MODULE      *
/*          BC1PNM60.                                              *
/*      THE FOLLOWING UPDATES MUST BE MADE TO THIS JCL BEFORE      *
/*      IT CAN BE EXECUTED:                                         *
/*      1. UPDATE THE JOBCARD TO REFLECT CORRECT SITE INFORMATION *
/*      2. REVIEW THAT THE SOURCE AND AUTHLIB DATA SET NAMES ARE   *
/*          CORRECT.                                               *
/*          - iprfix.iqual.SOURCE                                *
/*          - uprfix.uqual.AUTHLIB                               *
*/
*****  
//STEP1    EXEC PGM=ASMA90,  
// PARM='NODECK,OBJECT,NOTERM,LIST,XREF(SHORT),NOUSING,LANGUAGE(UE)'  
//SYSLIB   DD DISP=SHR,DSN=iprfix.iqual.SOURCE  
//          DD DISP=SHR,DSN=SYS1.MACLIB  
//SYSLIN   DD DISP=(NEW,PASS,DELETE),DSN=&&SYSLIN,  
//          UNIT=tdisk,SPACE=(TRK,(3,5),RLSE),  
//          DCB=(RECFM=FB,LRECL=80,BLKSIZE=3120)  
//SYSPUNCH DD DUMMY  
//SYSUT1   DD UNIT=tdisk,SPACE=(CYL,(5,3))  
//SYSPRINT DD SYSOUT=*  
//SYSIN    DD DISP=SHR,DSN=iprfix.iqual.SOURCE(BC1PNM60)  
*****  
/* BC1PNM60 SHOULD BE LINKED INTO YOUR SITE'S LOADLIB DATASET.  *
*****  
/*  
//STEP2    EXEC PGM=IEWL,PARM='LIST,NCAL,RENT,XREF,SIZE=(256K,64K)',  
//          COND=(0,NE)  
//SYSLIN   DD DISP=(OLD,DELETE),DSN=&&SYSLIN  
//SYSLMOD  DD DISP=SHR,DSN=uprfix.uqual.AUTHLIB(BC1PNM60)  
//SYSUT1   DD UNIT=tdisk,SPACE=(CYL,(5,3))  
//SYSPRINT DD SYSOUT=*
```

2.6 Trace Facility

2.6.1 EN\$TRNMI Trace

The Endevor CA-Netman trace facility validates that actions are working, allows you to see errors, shows you the information you're sending to CA-Netman, and displays the results of the actions.

To activate the CA-Netman internal trace facility during Endevor batch, jobs include the following JCL statement:

```
//EN$TRNMI DD      SYSOUT=A
```

To activate it during a TSO session, issue the following TSO allocation command before unlinking Endevor online:

```
ALLOC FI(EN$TRNMI) DSN('dataset name')
```

where '*dataset name*' is a trace data set with the following activities:

```
RECFM=FB  
LRECL=133
```

The trace data set must be sequential.

The following example was created after bringing up Endevor, resetting a package, and leaving Endevor:

(C) 1987,2000 Computer Associates International		Endevor for OS/390		09/18/00 07:58:58	PAGE 1		
ENDEVOR/OS/390 NETMAN INTERNAL TRACE							
BC1PNM00 EXT5RUTN: INITIALIZING VERSION 09/18/00 07:40							
BC1PNM00 EXT5RUTN: BC1TNM90 LOADED SUCCESSFULLY							
BC1PNM00 EXT5RUTN: SIGNON RC=00000							
FIND CHANGE RECORD							
00153FF8 (+0000)	C1E2D940 019000A0 00000000 00000000	C1F0F1C9 D5E3D4E2 40404040 40404040	*ASR A01INTMS *				
00154018 (+0020)	C5D5E3C5 D940C8E2 C5D35EC3 D3C5C1D9	5EE2C5E3 40E2C3E2 C6C9C5D3 C44DF15D	*ENTER HSEL;CLEAR;SET SCSFIELD(1)*				
00154038 (+0040)	7EC3C8D9 C5D8E3E8 D75EE2C5 E340E2C3	E2C6E5C1 D34DF15D 7ED5C4E5 D9D7D2C7	*=CHREQTYP;SET SCSFVAL(1)=NDVRPKG*				
00154058 (+0060)	5EE2C5E3 40E2C3E2 C6C9C5D3 C44DF25D	7EC3C8C3 C1E35EE2 C5E340E2 C3E2C6E5	*;SET SCSFIELD(2)=CHCAT;SET SCSFV*				
00154078 (+0080)	C1D34DF2 5D7E7DC1 D7D7D340 407D5EE2	C5E340E2 C3E2C3C8 C77E7DD5 C5E3D7D2	*AL(2)='APPL ',';SET SCSCHG='NETPK*				
00154098 (+00A0)	C77BF27D 5EE2C5E3 40E2C3E2 E3E2D27E	7D5C4040 40404040 407D5EC5 D5E3C5D9	*G#2';SET SCSTSK='* ' ;ENTER*				
BC1PNM10 PBFTRUTN: BC1PNM50 FINDCHG CALL RC=00 RE=000 ON RESET	BEFORE						
FIND CHANGE RECORD							
00153FF8 (+0000)	C1E2D940 019000A0 00000000 00000000	C1F0F1C9 D5E3D4E2 40404040 40404040	*ASR A01INTMS *				
00154018 (+0020)	C5D5E3C5 D940C8E2 C5D35EC3 D3C5C1D9	5EE2C5E3 40E2C3E2 C6C9C5D3 C44DF15D	*ENTER HSEL;CLEAR;SET SCSFIELD(1)*				
00154038 (+0040)	7EC3C8D9 C5D8E3E8 D75EE2C5 E340E2C3	E2C6E5C1 D34DF15D 7ED5C4E5 D9D7D2C7	*=CHREQTYP;SET SCSFVAL(1)=NDVRPKG*				
00154058 (+0060)	5EE2C5E3 40E2C3E2 C6C9C5D3 C44DF25D	7EC3C8C3 C1E35EE2 C5E340E2 C3E2C6E5	*;SET SCSFIELD(2)=CHCAT;SET SCSFV*				
00154078 (+0080)	C1D34DF2 5D7E7DC1 D7D7D340 407D5EE2	C5E340E2 C3E2C3C8 C77E7DD5 C5E3D7D2	*AL(2)='APPL ',';SET SCSCHG='NETPK*				
00154098 (+00A0)	C77BF27D 5EE2C5E3 40E2C3E2 E3E2D27E	7D5C4040 40404040 407D5EC5 D5E3C5D9	*G#2';SET SCSTSK='* ' ;ENTER*				
BC1PNM10 PAFTRUTN: BC1PNM50 FINDCHG CALL RC=00 RE=000 ON RESET	AFTER						
CREATE PACKAGE LOG RECORD							
00153FF8 (+0000)	C1E2D940 01900125 00000000 00000000	C1F0F1C9 D5E3D4E2 40404040 40404040	*ASR A01INTMS *				
00154018 (+0020)	C5D5E3C5 D940D6D7 D3C75EC3 D3C5C1D9	5EE2C5E3 40E2E8E2 E3C5D47E C3C8C75E	*ENTER OPLG;CLEAR;SET SYSTEM=CHG;*				
00154038 (+0040)	E2C5E340 D603D3D6 C37E7D5 C5E3D7D2	C77BF27D 5EE2C5E3 40D6D3C3 C1E37E7D	*SET OLLOC='NETPKG#2';SET OLCAT='*				
00154058 (+0060)	D5C47D5E E2C5E340 D603E3C5 E7E34DF1	5D7E7DD7 C1C3D2C1 C7C540D7 D9D6C3C5	*ND';SET OLTEXT(1)='PACKAGE PROCE*				
00154078 (+0080)	E2E2C9D5 C740C3D6 D4D7D3C5 E3C5C440	D9C340E7 40F04040 40404040 407D5EE2	*SSING COMPLETED RC = 0 'S*				
00154098 (+00A0)	C5E340D6 D3E3C5E7 E34DF25D 7E7DD5C5	E3D4C1D5 40C1C3C3 C5E2E2C5 C440C6D9	*ET OLTEXT(2)='NETMAN ACCESSED FR*				
00154088 (+00C0)	D6D440D7 C1C3D2C1 C7C540C5 E7C9E340	C1C6E3C5 D9404040 40D9C5E2 C5E34040	*OM PACKAGE EXIT AFTER RESET *				
001540D8 (+00E0)	407D5EE2 C5E340D6 D3E3C5E7 E34DF35D	7E7DD7D2 C740E2E3 C1E34E2 7A40C9D5	* ' ;SET OLTEXT(3)='PKG STATUS: IN*				
001540F8 (+0100)	60C5C4C9 E3404040 4040404E E2C5D97A	40D6D3C5 D1E4F0F1 407D5EE2 C5E340D6	*-EDIT USER: OLEJU01 ' ;SET 0*				
00154118 (+0120)	D3E3C5E7 E34DF45D 7E7DD9C5 E2C5E37A	40F1F8E2 C5D7F9F7 40F0F77A F5F97D5E	*LTEXT(4)='RESET: 18SEP97 07:59';*				
00154138 (+0140)	C5D5E3C5 D9		*ENTER*				
BC1PNM10 PAFTRUTN: BC1PNM50 CREATCLG CALL RC=00 RE=000 ON RESET	AFTER						
BC1PNM00 EXT6RUTN: SIGNOFF RC=00000							
BC1PNM00 EXT6RUTN: DELETED BC1TNM90							
BC1PNM00 EXT6RUTN: FREED ASR AND ARB BLOCKS							

Chapter 3. Setting Up the CA-Netman Interface

3.1 Overview

This chapter describes how to define Endevor action CCIDs and packages to CA-Netman.

3.2 Defining Endevor CCIDs to CA-Netman

To validate actions through CA-Netman, you need to set up a CA-Netman change record and task records. If your site elects not to validate Endevor CCIDs through CA-Netman, you do not need to define these records. They will be generated automatically when the Endevor interface creates log records.

3.2.1 CA-Netman Change Record

You must first create a CA-Netman change record for Endevor actions. This record is used to anchor task records which are used to identify Endevor action CCIDs. The following table describes change fields that are pertinent to the interface.

Change Field Name	Value	Description
Change id		CA-Netman will generate a unique ID. You may enter a value in this field.
Category	Category parameter value	Required by CA-Netman. Value will be set to equal the CATEGORY parameter value found in the User's Interface Definition table. This will only occur when Endevor creates this record.
Req Type	'ENDEVOR'	When the CA-Netman administrator creates this record, the value must be the same as defined in the User's Interface Definition table.
Requestor	Endevor user ID value	Required by CA-Netman. When Endevor creates the record, the field will equal Endevor's user ID.
Need by	'12/31/99'	Required by CA-Netman. When Endevor creates the record, the field will be set to 12/31/99.
Description	'ENDEVOR CCID CHANGE LIST'	Required by CA-Netman. When Endevor creates the record this field will be set to the following text: ENDEVOR CCID CHANGE LIST

3.2.1.1 Endevor Change Online View Example

The following is a sample of a CCID anchor change record.

```
CA-NETMAN (2) CHGC: Change Control ----- OLE 6/24/00 13:54
Command ==> _____ Seq
Seq
CHG
*****
Change Task Parent Category Prio Status Req Type Log Date
NDVR0001 APPL 00 ACTIVE ENDEVOR 6/20/00
Requestor Need By Dur Effort Resp Area
OLEJU01 12/31/00 _____
Desc ENDEVOR CCID CHANGE LIST. Log _____
_____
Prerequisite Tasks:
Sched Start End Approved Tested Install Backout Upd Rlse
Actual / / / / / / / / / / / / / / / /
Signed _____ _____ _____ _____ _____ _____ _____ _____
```

3.2.2 CA-Netman Task Records

CA-Netman task records must be defined for each Endevor action CCID if your site elects to validate CCIDs through CA-Netman. If not, these task records will be created during the interface's logging process.

The following table describes pertinent fields in a CA-Netman task record:

Change Field Name	Value	Description
Change id		Must be set to the Endevor change record ID value.
Task ID	Endevor CCID value	Required by the interface. This field must contain the Endevor CCID value. Since task records can only be 8 characters long, CCID values will be restricted to 8 characters.
Category	Category parameter value	Required by CA-Netman. Value will be set to equal the CATEGORY parameter value found in the User's Interface Definition table. This will only occur when Endevor creates this record. When the CA-Netman administrator creates this record, the value must be the same as defined in the User's Interface Definition table.

Change Field Name	Value	Description
Req Type	'ENDEVOR'	Required by the interface. Must be set to ENDEVOR.
Requestor	Endevor user ID value	Required by CA-Netman. When Endevor creates the record, the field will equal Endevor's user ID.
Need by	'12/31/00'	Required by CA-Netman. When Endevor creates the record, the field will be set to 12/31/00.
Description	'ENDEVOR CCID TASK RECORD'	Required by CA-Netman. When Endevor creates the record this field will be set to the following text: ENDEVOR CCID TASK RECORD
Log	Operlogid parameter value	Required by the interface. Value will be set to equal the OPERLOGID parameter value found in the User's Interface Definition table. This only occurs when Endevor creates this record. When the CA-Netman administrator creates this record, the value must be the same as defined in the User's Interface Definition table.

3.2.2.1 CA-Netman Task Log Records

As element activities occur, the interface creates log records and attaches them to the corresponding task record (by using the element's CCID value), if the startup definition log activity indicator is turned on. A description of how these log records are formatted follows.

Note: The element text line in each task log can continue for five lines when necessary to record long element names.

Through module BC1PNM60, the following four text lines are entered into the task log's comment area after the completion of an ADD, UPDATE, GENERATE, RETRIEVE, RESTORE, and DELETE element. If desired, module BC1PNM60 can be modified to change the log text.

ACTION PROCESSING COMPLETED RC=*action return code*
ELEMENT: *environment / system / subsystem / type / stage / element*
ACTION: *action* USER: *userid* DATE: *mm/dd/yy* TIME: *hh:mm*
COMMENT: *action comment*

The following five text lines are entered into the task log after the completion of an ARCHIVE element action:

ACTION PROCESSING COMPLETED RC=*action return code*
ELEMENT: *environment / system / subsystem / type / stage / element*
TARGET: *data set member*
ACTION: *action* USER: *userid* DATE: *mm/dd/yy* TIME: *hh:mm*
COMMENT: *action comment*

The following five text lines are entered into the task log after the completion of a MOVE and TRANSFER element action:

ACTION PROCESSING COMPLETED RC=*action return code*
SOURCE ELEMENT: *environment / system / subsystem / type / stage / element*
TARGET ELEMENT: *environment / system / subsystem / type / stage / element*
ACTION: *action* USER: *userid* DATE: *mm/dd/yy* TIME: *hh:mm*
COMMENT: *action comment*

Task log records are restricted to 12 lines. Because elements can be up to 255 characters, the target element name may be truncated. For example, if both the source and target element names equal 255 characters, the following lines will be recorded:

1. The action's return code statement
2. Six lines of source information
3. Three lines of target information. The target element name will have a '...' string on the third line to show that the name was truncated.
4. The action statement
5. The action's comment line

The following table describes the message text:

Message Text	Substituted with Endevor Value	Description
<i>action return code</i>	REQRTCOD	Action return code, obtained from REQRTCOD.
<i>environment</i>	Source ENVENV Target ENVENV	For Archive, Generate, Retrieve and Delete actions. Also for Move from location. For Add, Update and Restore actions. Also for Move to location.

Message Text	Substituted with Endevor Value	Description
<i>system</i>	Source ENVSYSTM	For Archive, Generate, Retrieve and Delete actions. Also for Move from location.
	Target ENVSYSTM	For Add, Update and Restore actions. Also for Move to location.
<i>subsystem</i>	Source ENVSUBSY	For Archive, Generate, Retrieve and Delete actions. Also for Move from location.
	Target ENVSUBSY	For Add, Update and Restore actions. Also for Move to location.
<i>type</i>	Source ENVTYPE	For Archive, Generate, Retrieve and Delete actions. Also for Move from location.
	Target ENVTYPE	For Add, Update and Restore actions. Also for Move to location.
<i>stage</i>	Source ENVSTGCD	For Archive, Generate, Retrieve and Delete actions. Also for Move from location.
	Target ENVSTGCD	For Add, Update and Restore actions. Also for Move to location.
<i>element</i>	Source ENVELEMNT	For Archive, Generate, Retrieve and Delete actions. Also for Move from location.
	Target ENVELEMNT	For Add, Update and Restore actions. Also for Move to location.
<i>action</i>	ECBFUNNAM	
<i>userid</i>	ECBUSER	
<i>mm/dd/yy</i>	Current date	date
<i>hh:mm</i>	Current time	time
<i>data set</i>	Target FILDSN	For Archive action.
<i>member</i>	Target FILDSMEM	For Archive action.
<i>action comment</i>	REQCOMM	

Note: Refer to the AllFusion Endevor Change Manager *Exits Guide* for more information.

3.2.2.2 Endevor Task Online View Example

The following is an output sample from a task record.

```
CA-NETMAN (2) CHGC: Change Control ----- OLE 6/24/00 13:45
Command ===> _____
Seq
CHG
*****
Change   Task      Parent      Category Prio Status  Req Type    Log Date
NDVR0001  CCID0002          APPL     00 ACTIVE  ENDEVOR  6/20/00
Requestor Need By Dur Effort
OLEJU01   12/31/00
Desc ENDEVOR CCID TASK RECORD._____
Log ND ACTION PROCESSING COMPLETED
_____
Prerequisite Tasks:
Start   End   Approved   Tested   Install   Backout   Upd Rlse
Sched   / /   / /   / /   / /   / /   / /   / /
Actual  / /   / /   / /   / /   / /   / /   / /
Signed  ____  ____  ____  ____  ____  ____  ____  ____
```

3.2.2.3 Endevor Task Log Online View Example

This sample displays the tracking events that occurred while adding an element for REL40T7:

```
CA-NETMAN (2) OPLG: Operation Log ----- OLE 11/01/02 10:49
Command ==> _____ Seq
LOG
*****
Log Date ID Cat Location System Type Unit Problem Date & ID
11/01/02 0003 ND NDVR0001 CHG REL40T7 _____ _/_/_____
Send To: _____ Action Date _/_/_ Received

Comments ACTION PROCESSING COMPLETED RC = 0
ELEMENT: I40 /NDVRMVS /JUDY /JDOHFS /1/JBEAN-JBEAN-JBEAN-
JBEAN-JBEAN-JBEAN-JBEAN-JBEAN-JBEAN-JBEAN-JBEAN-JBEAN-JB
EAN-JBEAN-JBEAN-JBEAN-JBEAN-JBEAN-JBEAN-JBEAN-JBEAN-JBEAN-
N-JBEAN-JBEAN-JBEAN-JBEAN-JBEAN-JBEAN-JBEAN-JBEAN-JBEAN-JBEA
N-JBEAN-JBEAN-JBEAN-JBEAN-JBEAN-JBEAN-JBEAN-JBEAN-JBEAN-JBEAN-
JBEAN-JBEAN-JBEAN-JBEAN-JBEAN-JBEAN-JBEAN-JBEAN-JBEAN-JBEAN-JB
EAN-JBEAN-JBEAN-JBEAN
ACTION: ADD USER: OLEJU01 DATE: 01NOV02 TIME: 10:49
COMMENT: NETMAN 255 ELEMENT NAME
_____
_____
```

==> _ Type C to set up a continuation entry for this log record.

3.3 Defining Endevor Packages to CA-Netman

If you want to validate packages through CA-Netman, you need to set up a CA-Netman change record for each package.

If your site elects NOT to validate Endevor packages through CA-Netman, you do not need to define records. They will be generated automatically when the Endevor interface is called to create a package log record. These log records will only be generated if logging is enabled for Endevor packages.

3.3.1 CA-Netman Change Record

The following table describes pertinent fields in a CA-Netman change record for package processing:

Change Field Name	Value	Description
Change id		Required by the interface. This field must be set with the Package name. Since CA-Netman change IDs can only be 8 characters long, Endevor package names will be restricted to 8 characters.
Category	Category parameter value	Required by CA-Netman. Value will be set to equal the CATEGORY parameter value found in the User's Interface Definition table. This will only occur when Endevor creates this record.
Req Type	'NDVRPKG'	When the CA-Netman administrator creates this record, the value must be the same as defined in the User's Interface Definition table.
Requestor	CA-Endevor user ID value	Required by CA-Netman. When Endevor creates the record, the field will equal Endevor's user ID.
Need by	'12/31/00'	Required by CA-Netman. When Endevor creates the record, the field will be set to 12/31/00.
Description	Endevor package comment	Required by CA-Netman. When Endevor creates the record, the field will equal Endevor's package comment value.

Change Field Name	Value	Description
Log	Operlogid parameter value	Required by the interface. Value will be set to equal the OPERLOGID parameter value found in the User's Interface Definition table. This only occurs when Endevor creates this record. When the CA-Netman administrator creates this record, the value must be the same as defined in the User's Interface Definition table.

3.3.1.1 CA-Netman Change Log Record

As element activities occur, the interface creates log records and attaches them to the corresponding task record (by using the package ID value), if the startup definition log activity indicator is turned on.

Through module BC1PNM60, the following text lines are entered into the change log's comment area. If desired, module BC1PNM60 can be modified to change the log text.

On completion of a package CREATE and MODIFY function:

PACKAGE PROCESSING COMPLETED RC=*package return code*
NETMAN ACCESSED FROM PACKAGE EXIT *exit/package function*
PKG STATUS: *status* USER: *userid*
CREATE: *package create date and time*

On completion of a package CAST function:

PACKAGE PROCESSING COMPLETED RC=*package return code*
NETMAN ACCESSED FROM PACKAGE EXIT *exit/package function*
PKG STATUS: *status* USER: *userid*
CAST: *package cast date and time*

On completion of a package REVIEW function:

PACKAGE PROCESSING COMPLETED RC=*package return code*
NETMAN ACCESSED FROM PACKAGE EXIT *exit/package function*
PKG STATUS: *status* USER: *userid*

On completion of a package DELETE function:

PACKAGE PROCESSING COMPLETED RC=*package return code*
NETMAN ACCESSED FROM PACKAGE EXIT *exit/package function*
PKG STATUS: N/A USER: *userid*

On completion of a package EXECUTE function:

PACKAGE PROCESSING COMPLETED RC=*package return code*
NETMAN ACCESSED FROM PACKAGE EXIT *exit/package function*
PKG STATUS: *status* USER: *userid*
EXECUTION: *execution start date and time*
END EXECUTION: *execution end date and time*

On completion of a package BACKOUT function:

PACKAGE PROCESSING COMPLETED RC=*package return code*
NETMAN ACCESSED FROM PACKAGE EXIT *exit/package function*
PKG STATUS: *status* USER: *userid*
BACKOUT: *backout date and time* STATUS: *backout status*

On completion of a package BACKIN function:

PACKAGE PROCESSING COMPLETED RC=*package return code*
NETMAN ACCESSED FROM PACKAGE EXIT *exit/package function*
PKG STATUS: *status* USER: *userid*
BACKIN: *backin date and time*

On completion of a package COMMIT function:

PACKAGE PROCESSING COMPLETED RC=*package return code*
NETMAN ACCESSED FROM PACKAGE EXIT *exit/package function*
PKG STATUS: *status* USER: *userid*
COMMIT: *commit date and time*

On completion of a package RESET function:

PACKAGE PROCESSING COMPLETED RC=*package return code*
NETMAN ACCESSED FROM PACKAGE EXIT *exit/package function*
PKG STATUS: *status* USER: *userid*
RESET: *current date and time*

On completion of a package SHIP XMIT function:

PACKAGE PROCESSING COMPLETED RC=*package return code*
NETMAN ACCESSED FROM PACKAGE EXIT *exit/package function*
PKG STATUS: *status* USER: *userid*
SHIP: DESTINATION *ship destination* TYPE *shipment type* COMPLEMENTS *y/n*

On completion of a package SHIP CONFIRM function:

PACKAGE PROCESSING COMPLETED RC=*package return code*
NETMAN ACCESSED FROM PACKAGE EXIT *exit/package function*
PKG STATUS: *status* USER: *userid*
SHIP: DESTINATION *ship destination* TYPE *confirm type* RESULTS *results and rc*

The following table describes the message text:

Message Text	Substituted with Endevor Value	Description
<i>package return code</i>	PECBNDRC	Endevor high return code
<i>exit/package function</i>	PECBBANM PECBFNNM	Either BEFORE or AFTER Package function
<i>status</i>	PHDRSTAT	Package status
<i>userid</i>	PECBUSER	User ID
<i>package create date and time</i>	PHDRCRD PHDRCRT	Package create date Package create time
<i>package cast date and time</i>	PHDRCD PHDRCT	Package cast date Package cast time

Message Text	Substituted with Endevor Value	Description
<i>execution start date and time</i>	PHDRXD PHDRXT	Package start execution date Package start execution time
<i>execution end date and time</i>	PHDREEXD PHDREEXT	Package end execution date Package end execution time
<i>backout date and time</i>	PHDRBOD PHDRBOT	Package backout date Package backout time
<i>backout status</i>	PHDRBOST	Package backout status
<i>backin date and time</i>	PHDRBID PHDRBIT	Package backin date Package backin time
<i>commit date and time</i>	PHDRCMD PHDRCMT	Package commit date Package commit time
<i>current date and time</i>	Current date Current time	ddmmmyy hh:mm
<i>ship destination</i>	PREQDEST	Package ship destination
<i>shipment type</i>	PREQSTYP	Package shipment type
<i>y/n</i>	PREQSCMP	Ship complements (either y or n)
<i>confirm type</i>	PREQSCNF	Ship confirmation type
<i>results and rc</i>	PREQSRES PREQSRCV	Ship confirmation results Ship confirm return code

3.3.1.2 Endevor Change Online View Example

The following is a sample of a package change record.

CHG								
Change	Task	Parent	Category	Prio	Status	Req Type	Log Date	
PKGJ0001			APPL	00	ACTIVE	NDVRPKG	6/20/00	
Requestor	Need By	Dur	Effort					Resp Area
OLEJU01	12/31/00							
Desc	PACKAGE DESCRIPTION	FROM	NDVR		Log	ND	PACKAGE PROCESSING	COMPLETED
<hr/>								
Prerequisite Tasks:								
Sched	/ /	/ /	/ /	/ /	/ /	/ /	/ /	/ /
Actual	/ / /	/ / /	/ / /	/ / /	/ / /	/ / /	/ / /	/ / /
Signed	—	—	—	—	—	—	—	—

3.3.1.3 Endevor Change Log Online View Example

This sample displays the tracking events that occurred during package processing:

```
CA-NETMAN (2) OPLG: Operation Log ----- OLE 6/20/00 14:46
Command ==> _____ Seq
LOG
*****
Log Date ID Cat Location System Type Unit Problem Date & ID
6/20/00 0008 ND PKGJ0001 CHG _____ _/_/
Send To: _____ Action Date _/_/_ Received
Comments PACKAGE PROCESSING COMPLETED RC=0
NETMAN ACCESSED FROM PACKAGE EXIT AFTER/CREATE
PKG STATUS: IN-EDIT USER: OLEJU01A
CREATE: 06/20/00 12:20
_____
_____
_____
_____
_____
MORE...
==> _ Type C to set up a continuation entry for this log record.
```

Chapter 4. Verifying Endevor Activities Against CA-Netman

4.1 Overview

This chapter describes how to verify Endevor activities against CA-Netman using action processing and package processing.

4.2 Element Action Processing

The CA-Netman Interface uses CA-Netman's task records as a repository for all Endevor CCIDs. (A separate task record is used for a single CCID.) A CA-Netman change record is used to anchor all the task records. These records may be created on either the CA-Netman or Endevor side (based on the startup definition CCID validation indicator). The task record identification field is set to the element's CCID value. This limits the CCID value to eight characters. As element activities occur, CA-Netman log records are created and attached to the corresponding task record (by using the element's CCID value) if the startup definition log activity indicator is turned on.

When CA-Netman errors occur, the interface simply returns to Endevor, allowing Endevor processing to continue normally if the interface startup definition "Netman Required" indicator is set to NO. If the indicator is set to YES, the interface will cancel the Endevor action during exit 2 (Before Action) (Print and List element actions are NOT affected).

The Before (exit 2) and After (exit 3) exits operate for the following Endevor actions: Add, Update, Restore, Retrieve, Archive, Generate, Delete, Move, and Transfer.

Note: Any Endevor field or text string that uses the underscore (_) character appears as a blank in Netman. The underscore character is valid for use in an Endevor HFS element name.

4.2.1 Before Element Action

During exit 2, CCID validation is done. This interface will behave differently based on the interface startup definition "CCID Validation" indicator. When the flag is turned on, the interface checks for a CCID task record. If found, the CCID is deemed valid. If not found, CCID validation is done by Endevor through C1GSCIPO which is another exit 2 module. When the flag is turned off, CCID validation is only done by Endevor through C1GSCIPO.

4.3 Package Processing

Package information is stored on change records. Package activity is recorded on log records associated to the package change record.

Note: Package element action activity is still recorded on log records associated to the task (CCID) record which in turn is anchored off Endevor's repository change record.

All package change records may be created on either the CA-Netman or Endevor side (depending on the interface's startup definition table "Package Validation" indicator). The change record identification field is set to the package name, thus limiting the package name to eight characters.

When CA-Netman errors occur, the interface simply returns to Endevor, allowing Endevor processing to continue normally if the interface startup definition "Netman Required" indicator is set to NO. If the indicator is set to YES, the interface cancels the package function during exit 7 (Before Package).

The interface utilizes the Before and After (exit 7) exits on the following Endevor package functions: Backin, Backout, Cast, Commit, Confirm, Create, Delete, Execute, Modify, Reset, Review, and Ship.

4.3.1 Before Package Function

During exit 7, package validation is done. The interface behaves differently based on the startup definition "PKG Validation" indicator. When the flag is turned on, the interface checks for a package change record. If found, the package is deemed valid. If not found, the package function is canceled. When the flag is turned off, the package change record is created if one does not already exist.

Chapter 5. Logging Endevor Activities in CA-Netman

5.1 Overview

This chapter describes how to log Endevor activities within CA-Netman.

5.2 After Element Action

During exit 3 (After Action), log records are created and attached to the proper task record only if the startup definition log activity indicator is turned on. If the task record does not exist, the interface will create it under two conditions:

- When the action CCID is deemed valid.
- The validation indicator is turned off.

Note: If logging is not turned on, a task record is not created by Endevor. When logging is enabled and the Endevor change record is not defined within CA-Netman's database at the time a task record is to be created, the interface will create the Endevor change record.

5.3 After Package Function

During exit 7 (After Package), the startup definition package log indicator is examined. If the indicator is turned on, log records are created and attached to the proper package change record.

5.4 Modifying Log Information

5.4.1 CA-Netman Record Population

During the CA-Netman record population process for change, task, and log records, module BC1PNM60 is called to build the NETMAN API command string to populate these record fields.

Note: Refer to Appendix A, "BC1PNM60 Source Module," for a display of BC1PNM60.

After you change the program, it should be assembled and linked into your site's loadlib data set. This loadlib should be concatenated in Endevor's steplib. The original BC1PNM60 load module can be found in the Endevor data set, iprfix.iqual.SOURCE.

The following displays sample JCL for assembling and linking module BC1PNM60:

```
/* ( COPY JOBCARD )
//***** *****
//*
//*      BC1JNM60 - THIS JOB IS USED TO ASSEMBLE AND LINK THE *
//*                  ENDEVOR-MVS NETMAN INTERFACE POPULATION MODULE *
//*                  BC1PNM60. *
//*
//*      THE FOLLOWING UPDATES MUST BE MADE TO THIS JCL BEFORE *
//*      IT CAN BE EXECUTED: *
//*
//*      1. UPDATE THE JOBCARD TO REFLECT CORRECT SITE INFORMATION *
//*      2. REVIEW THAT THE SOURCE AND LOADLIB DATA SET NAMES ARE *
//*          CORRECT. *
//*              - iprfix.iqual.SOURCE *
//*              - uprfix.uqual.LOADLIB *
//*
//***** *****
//STEP1    EXEC PGM=ASMA90,
// PARM='NODECK,OBJECT,NOTERM,LIST,XREF(SHORT),NOUSING,LANGUAGE(UE)'
//SYSLIB   DD  DISP=SHR,DSN=iprfix.iqual.SOURCE
//         DD  DISP=SHR,DSN=SYS1.MACLIB
//SYSLIN   DD  DISP=(NEW,PASS,DELETE),DSN=&&SYSLIN,
//         UNIT=tdisk,SPACE=(TRK,(3,5),RLSE),
//         DCB=(RECFM=FB,LRECL=80,BLKSIZE=3120)
//SYSPUNCH DD  DUMMY
//SYSUT1   DD  UNIT=tdisk,SPACE=(CYL,(5,3))
//SYSPRINT DD  SYSOUT=*
//SYSIN    DD  DISP=SHR,DSN=iprfix.iqual.SOURCE(BC1PNM60)
//***** *****
//* BC1PNM60 SHOULD BE LINKED INTO YOUR SITE'S LOADLIB DATASET. *
//***** *****
//*
//STEP2    EXEC PGM=IEWL,PARM='LIST,NCAL,RENT,XREF,SIZE=(256K,64K)',
//           COND=(0,NE)
//SYSLIN   DD  DISP=(OLD,DELETE),DSN=&&SYSLIN
//SYSLMOD  DD  DISP=SHR,DSN=uprfix.uqual.LOADLIB(BC1PNM60)
//SYSUT1   DD  UNIT=tdisk,SPACE=(CYL,(5,3))
//SYSPRINT DD  SYSOUT=*
```

Appendix A. BC1PNM60 Source Module

A.1 BC1PNM60

This appendix displays the BC1PNM60 source module.

```
BC1PNM60 CSECT
BC1PNM60 AMODE 31
BC1PNM60 RMODE ANY
    SPACE 1
*****
*
* NAME:      BC1PNM60
*
* ROUTINES:   MAIN      - MAINLINE ROUTINE.
*              CTSKRUTN - SET FIELD VALUES FOR TASK RECORD.
*              CCHARUTN - SET FIELD VALUES FOR ENDEVOR CHANGE
*                           RECORD.
*              CCHPRUTN - SET FIELD VALUES FOR PACKAGE CHANGE
*                           RECORD.
*              CTLGRUTN - SET FIELD VALUES FOR TASK LOG RECORD.
*              CCLGRUTN - SET FIELD VALUES FOR CHANGE LOG
*                           RECORD.
*
* SPACE 1
*****
* CUSTOMER NETMAN DEFINITION TABLE
*****
@ENETM DSECT
EJECT
*****
* EXIT CONTROL BLOCK
*****
$ECBDS
EJECT
*****
* REQUEST INFORMATION BLOCK
*****
$REQPDS
EJECT
*****
* ENVIRONMENT BLOCK
*****
$ENVDS
EJECT
```

```
*****
* FILE BLOCK
*****
$FIELDS
EJECT
*****
* PACKAGE EXIT CONTROL BLOCK STRUCTURE
*****
$PECBDS
EJECT
*****
* PACKAGE EXIT HEADER CONTROL BLOCK STRUCTURE
*****
$PHDRDS
EJECT
*****
* PACKAGE EXIT REQUEST CONTROL BLOCK STRUCTURE
*****
$PREQPD$S
EJECT
*****
* REGISTER EQUATES
*****
R0    EQU   0           REGISTER 0
R1    EQU   1           REGISTER 1
R2    EQU   2           REGISTER 2
R3    EQU   3           REGISTER 3
R4    EQU   4           REGISTER 4
R5    EQU   5           REGISTER 5
R6    EQU   6           REGISTER 6
R7    EQU   7           REGISTER 7
R8    EQU   8           REGISTER 8
R9    EQU   9           REGISTER 9
R10   EQU   10          REGISTER 10
R11   EQU   11          REGISTER 11
R12   EQU   12          REGISTER 12
R13   EQU   13          REGISTER 13
R14   EQU   14          REGISTER 14
R15   EQU   15          REGISTER 15
EJECT
```

```
*****
* BC1PNM60 REENTRANT WORK AREA STRUCTURE. *
*****
WA1DSECT DSECT
    WA1_MAINSAVE    DS   18F      MAIN REGISTER SAVE AREA.
    WA1_CTSKSAVE    DS   15F      CTSKRUTN REGISTER SAVE AREA.
    WA1_CCHASAVE    DS   15F      CCHARUTN REGISTER SAVE AREA.
    WA1_CCHPSAVE    DS   15F      CCHPRUTN REGISTER SAVE AREA.
    WA1_CTLGSAVE    DS   15F      CTLGRUTN REGISTER SAVE AREA.
    WA1_CCLGSAVE    DS   15F      CCLGRUTN REGISTER SAVE AREA.
        SPACE 1
    WA1_BUFF_LEN     DS   H       CALLER'S BUFFER LENGTH.
    WA1_BUFF_ADDR    DS   F       CALLER'S BUFFER ADDRESS.
    WA1_DATALEN_ADDR DS   F       ADDRESS OF CALLER'S HW AREA.
    *
    WA1_CURR_DATE   DS   CL7      DDMMYY CURRENT DATE.
    WA1_CURR_TIME   DS   CL5      HH:MM   CURRENT TIME.
        SPACE 1
    * WORK FIELDS FOR DATA CONVERSION
    WA1_DBL         DS   D       WORK - DOUBLE WORD.
    WA1_TARGET1      DS   CL8      WORK - DISPLAY FORM.
    WA1_TARGET2      DS   CL8      WORK - DISPLAY FORM.
        SPACE 1
    WA1LEN EQU     *-WA1DSECT
    EJECT
    TITLE 'BC1PNM60 - MAINLINE'
*****
* BC1PNM60 - MAIN
*
* REGISTER USAGE:
*   R12:      BASE REGISTERS
*   R13:      WA1DSECT
*   R9:       ENETM
*   R8:       EXIT C/B ANCHOR
*
*****
```

```

BC1PNM60 CSECT
    B    MAIN-BC1PNM60(,R15)
    DC   C'BC1PNM60 '
    DC   C'&SYSDATE '
    DC   C'&SYSTIME '
    SPACE 1
MAIN    DS    0H
        STM   R14,R12,12(R13)
        LR    R12,R15
        USING BC1PNM60,R12
        SPACE 1
        LR    R6,R1
        GETMAIN RC, LV=WA1LEN
        LR    R14,R1
        LA    R15,WA1LEN
        XR    R3,R3
        MVCL  R14,R2
        ST    R1,8(,R13)
        ST    R13,4(,R1)
        LR    R13,R1
        USING WA1DSECT,R13
        SPACE 1
        MVC   WA1_BUFF_ADDR,4(R6)      SAVE CALLER'S BUF ADDR IN WA1
        L     R5,8(,R6)                R5 -> CALLER'S BUFFER LEN
        MVC   WA1_BUFF_LEN,0(R5)       SAVE BUFFER LENGTH VALUE IN WA1
        MVC   WA1_DATALEN_ADDR,12(R6)  SAVE ADDR OF WORK HALFWORD -
                                         USED TO UPDT W/ LEN.
*
        L     R9,16(,R6)              ESTABLISH R9 FOR ENETM -
                                         USER DEFINITION TABLE.
        USING ENETM,R9
        L     R8,20(,R6)              R8-> ENDEVOR EXIT C/B ANCHOR
        L     R1,24(,R6)
        MVC   WA1_CURR_DATE,0(R1)    GET CURR DATE.
        L     R1,28(,R6)
        MVC   WA1_CURR_TIME,0(R1)    GET CURR TIME.
        L     R6,0(,R6)               R6->FUNCTION CODE.
        SPACE 1
*****
* PERFORM CTSKRUTN WHEN FUNCTION = 'CTSK'. *
* ADD CHANGE FIELD VALUES DURING ENDEVOR'S CREATE NETMAN TASK *
* RECORD FUNCTION. *
*****
        CLC   0(4,R6),=CL4'CTSK'
        BNE  MAIN0020
        L    R15,=A(CTSKRUTN)
        BALR R14,R15
        B    MAIN0900

```

```
*****
* PERFORM CCHARUTN WHEN FUNCTION = 'CCHA'. *
*   ADD CHANGE FIELD VALUES DURING ENDEVOR'S CREATE NETMAN CHANGE   *
*   RECORD DURING ACTION PROCESSING. THIS RECORD ACTS AS AN ANCHOR   *
*   TO ALL ENDEVOR DEFINED TASKS.                                     *
*****
MAIN0020 DS    OH
    CLC  O(4,R6),=CL4'CCHA'
    BNE  MAIN0030
    L    R15,=A(CCHARUTN)
    BALR R14,R15
    B    MAIN0900
    EJECT
*****
* PERFORM CCHPRUTN WHEN FUNCTION = 'CCHP'. *
*   ADD CHANGE FIELD VALUES DURING ENDEVOR'S CREATE NETMAN CHANGE   *
*   RECORD FOR PACKAGE ACTIONS.                                     *
*****
MAIN0030 DS    OH
    CLC  O(4,R6),=CL8'CCHP'
    BNE  MAIN0040
    L    R15,=A(CCHPRUTN)
    BALR R14,R15
    B    MAIN0900
    EJECT
*****
* PERFORM CTLGRUTN WHEN FUNCTION = 'CTLG'. *
*   ADD LOG TEXT VALUES DURING ENDEVOR'S CREATE NETMAN TASK LOG   *
*   RECORD FOR ELEMENT ACTIONS.                                     *
*****
MAIN0040 DS    OH
    CLC  O(4,R6),=CL4'CTLG'
    BNE  MAIN0050
    L    R15,=A(CTLGRUTN)
    BALR R14,R15
    B    MAIN0900
    EJECT
```

```

*****
* PERFORM CCLGRUTN WHEN FUNCTION = 'CCLG'. *
* ADD LOG TEXT VALUES DURING ENDEVOR'S CREATE NETMAN CHANGE LOG   *
* RECORD FOR PACKAGE ACTIONS.                                     *
*****
MAIN0050 DS    OH
    CLC  0(4,R6),=CL4'CCLG'
    BNE  MAIN0900
    L    R15,=A(CCLGRUTN)
    BALR R14,R15
    B    MAIN0900
    EJECT
MAIN0900 DS    OH
    LR   R1,R13           SET R1 TO WA1DSECT.
    L    R13,4,(R13)        RESTORE OLD SAVE AREA.
    FREEMAIN RC,A=(R1),LV=WA1LEN
    LM   R14,R12,12(R13)
    XR   R15,R15
    BR   R14
    SPACE 1
    LTORG
    DROP  R12
    EJECT
    TITLE 'BC1PNM60 -ADD CHANGE VALUES FOR A TASK RECORD'
*****
* BC1PNM60 - CTSK
* AT THIS POINT, THE API BUFFER CONTAINS THE "CHGC" CMD FOLLOWED   *
* BY THE "CLEAR" CMD. THE CHCHANGE, CHTASK AND CHREQTYP VALUES      *
* ARE ALREADY PLACED WITHIN THE BUFFER.                                *
*
* THIS ROUTINE WILL ADD THE FOLLOWING "SET" CMDS.                   *
* SET CHCAT='ENMCATGY';          WHERE ENMCATGY REPRESENTS THE     *
*                               CONTAINS OF THE FIELD.                  *
* SET CHREQSTR='ECBUSER';        WHERE ECBUSER REPRESENTS THE     *
*                               CONTAINS OF THE FIELD.                  *
* SET CHLOGID='ENMLOGID';        WHERE ENMLOGID REPRESENTS THE     *
*                               CONTAINS OF THE FIELD.                  *
* SET CHNEEDBY='12/31/10';       *
* SET CHDESC='ENDEVOR CCID TASK RECORD';                            *
*
* REGISTER USAGE:                                                 *
* R2-R4: WORK          *
* R8:    EXIT CONTROL BLOCK ANCHOR   *
* R9:    ADDR OF ENETM STRUCTURE (USER DEFINITION TABLE)   *
* R12:   CTSKRUTN BASE ADDRESS   *
* R13:   WA1DSECT STRUCTURE ADDRESS   *
*****

```

```

CTSKRUTN DS   0H
    STM R12,R14,WA1_CTSKSAVE
    LR  R12,R15
    USING CTSKRUTN,R12
    SPACE 1
* COMPUTE TOTAL STRING LENGTH.
    LA  R2,L'CTSK_L1+L'CTSK_L2+L'CTSK_L3+L'CTSK_L4+L'CTSK_L5
    LA  R2,L'ENMCATGY+L'ENMLOGID+L'ECBUSER(,R2)
    CH  R2,WA1_BUFF_LEN
    BH  CTSKEXIT
    L  R4,WA1_DATALEN_ADDR      UPDATE CALLER'S AREA WITH
    STH R2,0(,R4)                TOTAL DATA LENGTH.
    SPACE 1
* PLACE 1ST SET STRING INTO BUFFER.
    L  R3,WA1_BUFF_ADDR        R3 -> BUFFER
    MVC 0(L'CTSK_L1,R3),CTSK_L1  MOVE STRING 1 INTO BUFFER
    LA  R3,L'CTSK_L1(,R3)       BUMP BUFFER BY STR LEN
    MVC 0(L'ENMCATGY,R3),ENMCATGY  MOVE IN VARIABLE
    LA  R3,L'ENMCATGY(,R3)      BUMP BUFFER BY VAR LEN
    SPACE 1
* PLACE 2ST SET STRING INTO BUFFER.
    MVC 0(L'CTSK_L2,R3),CTSK_L2  MOVE STRING 2 INTO BUFFER
    LA  R3,L'CTSK_L2(,R3)       BUMP BUFFER BY STR LEN
    L  R4,0(,R8)               R4 -> $ECBDS C/B
    USING $ECBDS,R4
    MVC 0(L'ECBUSER,R3),ECBUSER  MOVE IN VARIABLE
    LA  R3,L'ECBUSER(,R3)       BUMP BUFFER BY VAR LEN
    DROP R4
    SPACE 1
    MVC 0(L'CTSK_L3,R3),CTSK_L3  MOVE STRING 3 INTO BUFFER
    LA  R3,L'CTSK_L3(,R3)       BUMP BUFFER BY STR LEN
    MVC 0(L'ENMLOGID,R3),ENMLOGID  MOVE IN VARIABLE
    LA  R3,L'ENMLOGID(,R3)      BUMP BUFFER BY VAR LEN
    SPACE 1
    MVC 0(L'CTSK_L4,R3),CTSK_L4  MOVE STRING 4 INTO BUFFER
    LA  R3,L'CTSK_L4(,R3)       BUMP BUFFER BY STR LEN
    SPACE 1
    MVC 0(L'CTSK_L5,R3),CTSK_L5  MOVE STRING 5 INTO BUFFER
    LA  R3,L'CTSK_L5(,R3)       BUMP BUFFER BY STR LEN
    SPACE 1
CTSKEXIT DS   0H
    LM  R12,R14,WA1_CTSKSAVE
    XR  R15,R15
    BR  R14
    SPACE 1

```

```
*****
* CTSK - CONSTANTS
*****
CTSK_L1 DC C'SET CHCAT=''''
CTSK_L2 DC C'''';SET CHREQSTR=''''
CTSK_L3 DC C'''';SET CHLOGID=''''
CTSK_L4 DC C'''';SET CHNEEDBY='12/31/10''';
CTSK_L5 DC C'SET CHSDESC='ENDEVOR CCID TASK RECORD''';
LTORG
DROP R12
EJECT
TITLE 'BC1PNM60 - ADD CHANGE VALUES FOR AN ELEMENT ACTION'
*****
* BC1PNM60 - CCHA
* AT THIS POINT, THE API BUFFER CONTAINS THE "CHGC" CMD FOLLOWED *
* BY THE "CLEAR" CMD. THE CHREQTYP VALUE IS ALREADY PLACED IN THE *
* BUFFER. *
*
* THIS ROUTINE WILL ADD THE FOLLOWING "SET" CMDS.
* SET CHCAT='ENMCATGY'; WHERE ENMCATGY REPRESENTS THE *
* CONTAINS OF THE FIELD.
* SET CHREQSTR='ECBUSER'; WHERE ECBUSER REPRESENTS THE *
* CONTAINS OF THE FIELD.
* SET CHNEEDBY='12/31/10';
* SET CHSDESC='ENDEVOR CCID CHANGE LIST';
*
* REGISTER USAGE:
* R2-R4: WORK
* R8: EXIT CONTROL BLOCK ANCHOR
* R9: ADDR OF ENETM STRUCTURE (USER DEFINITION TABLE)
* R12: CCHARUTN BASE ADDRESS
* R13: WA1DSECT STRUCTURE ADDRESS
*
*****
```

```

CCHARUTN DS    0H
    STM   R12,R14,WA1_CCHASAVE
    LR    R12,R15
    USING CCHARUTN,R12
    SPACE 1
* COMPUTE TOTAL STRING LENGTH.
    LA    R2,L'CCHA_L1+L'CCHA_L2+L'CCHA_L3+L'CCHA_L4
    LA    R2,L'ENMCATGY+L'ECBUSER(,R2)
    CH    R2,WA1_BUFF_LEN
    BH    CCHAEXIT
    L    R4,WA1_DATALEN_ADDR      UPDATE CALLER'S AREA WITH
    STH   R2,0(,R4)              TOTAL DATA LENGTH.
    SPACE 1
* PLACE 1ST SET STRING INTO BUFFER.
    L    R3,WA1_BUFF_ADDR        R3 -> BUFFER
    MVC  0(L'CCHA_L1,R3),CCHA_L1  MOVE STRING 1 INTO BUFFER
    LA    R3,L'CCHA_L1(,R3)      BUMP BUFFER BY STR LEN
    MVC  0(L'ENMCATGY,R3),ENMCATGY MOVE IN VARIABLE
    LA    R3,L'ENMCATGY(,R3)      BUMP BUFFER BY VAR LEN
    SPACE 1
* PLACE 2ST SET STRING INTO BUFFER.
    MVC  0(L'CCHA_L2,R3),CCHA_L2  MOVE STRING 2 INTO BUFFER
    LA    R3,L'CCHA_L2(,R3)      BUMP BUFFER BY STR LEN
    L    R4,0(,R8)                R4 -> $ECBDS C/B
    USING $ECBDS,R4
    MVC  0(L'ECBUSER,R3),ECBUSER  MOVE IN VARIABLE
    LA    R3,L'ECBUSER(,R3)      BUMP BUFFER BY VAR LEN
    DROP  R4
    SPACE 1
    MVC  0(L'CCHA_L3,R3),CCHA_L3  MOVE STRING 3 INTO BUFFER
    LA    R3,L'CCHA_L3(,R3)      BUMP BUFFER BY STR LEN
    SPACE 1
    MVC  0(L'CCHA_L4,R3),CCHA_L4  MOVE STRING 4 INTO BUFFER
    LA    R3,L'CCHA_L4(,R3)      BUMP BUFFER BY STR LEN
    SPACE 1
CCHAEXIT DS    0H
    LM    R12,R14,WA1_CCHASAVE
    XR    R15,R15
    BR    R14
    SPACE 1
*****
* CCHA - CONSTANTS
*****
CCHA_L1 DC    C'SET CHCAT=''
CCHA_L2 DC    C''';SET CHREQSTR=''
CCHA_L3 DC    C''';SET CHNEEDBY='12/31/10'';;
CCHA_L4 DC    C'SET CHSDESC='ENDEVOR CCID CHANGE LIST'';
LTORG
DROP   R12
EJECT
TITLE 'BC1PNM60 -ADD CHANGE VALUES FOR A PACKAGE ACTION'

```

```
*****
* BC1PNM60 - CCHP
* AT THIS POINT, THE API BUFFER CONTAINS THE "CHGC" CMD FOLLOWED
* BY THE "CLEAR" CMD. THE CHCHANGE, CHREQTYP AND CHREQSTR "SET"
* COMMANDS ARE ALREADY PLACED IN THE BUFFER.
*
* THIS ROUTINE WILL ADD THE FOLLOWING "SET" CMDS.
* SET CHCAT='ENMCATGY'; WHERE ENMCATGY REPRESENTS THE
* CONTAINS OF THE FIELD.
* SET CHLOGID='ENMLOGID'; WHERE ENMLOGID REPRESENTS THE
* CONTAINS OF THE FIELD.
* SET CHSDESC='PREQCOMM'; WHERE PREQCOMM REPRESENTS THE
* CONTAINS OF THE 1ST 36 CHARACTERS
* OF THE FIELD.
* SET CHNEEDBY='12/31/10';
*
* REGISTER USAGE:
* R2-R4: WORK
* R8: EXIT CONTROL BLOCK ANCHOR
* R9: ADDR OF ENETM STRUCTURE (USER DEFINITION TABLE)
* R12: CCHPRUTN BASE ADDRESS
* R13: WA1DSECT STRUCTURE ADDRESS
*****
CCHPRUTN DS    0H
      STM   R12,R14,WA1_CCHPSAVE
      LR    R12,R15
      USING CCHPRUTN,R12
      SPACE 1
* COMPUTE TOTAL STRING LENGTH.
      LA    R2,L'CCHP_L1+L'CCHP_L2+L'CCHP_L3+L'CCHP_L4
      LA    R2,L'ENMCATGY+L'ENMLOGID+36(,R2)
      CH    R2,WA1_BUFF_LEN
      BH    CCHPEXIT
      L     R4,WA1_DATALEN_ADDR      UPDTE CALLER'S AREA WITH
      STH   R2,0(,R4)                TOTAL DATA LENGTH.
      SPACE 1
* PLACE 1ST SET STRING INTO BUFFER.
      L     R3,WA1_BUFF_ADDR        R3 -> BUFFER
      MVC  0(L'CCHP_L1,R3),CCHP_L1  MOVE STRING 1 INTO BUFFER
      LA    R3,L'CCHP_L1(,R3)       BUMP BUFFER BY STR LEN
      MVC  0(L'ENMCATGY,R3),ENMCATGY MOVE IN VARIABLE
      LA    R3,L'ENMCATGY(,R3)      BUMP BUFFER BY VAR LEN
      SPACE 1
```

```

* PLACE 2ND SET STRING INTO BUFFER.
  MVC 0(L'CCHP_L2,R3),CCHP_L2      MOVE STRING 2 INTO BUFFER
  LA   R3,L'CCHP_L2(,R3)          BUMP BUFFER BY STR LEN
  MVC 0(L'ENMLOGID,R3),ENMLOGID    MOVE IN VARIABLE
  LA   R3,L'ENMLOGID(,R3)          BUMP BUFFER BY VAR LEN
  SPACE 1
  MVC 0(L'CCHP_L3,R3),CCHP_L3      MOVE STRING 3 INTO BUFFER
  LA   R3,L'CCHP_L3(,R3)          BUMP BUFFER BY STR LEN
  L   R4,4(,R8)                  R4-> $PREQPDS C/B
  USING $PREQPDS,R4
  MVC 0(36,R3),PREQCOMM          MOVE IN VARIABLE (1ST 36)
  LA   R3,36(,R3)                BUMP BUFFER BY 36
  DROP R4
  SPACE 1
  MVC 0(L'CCHP_L4,R3),CCHP_L4      MOVE STRING 4 INTO BUFFER
  LA   R3,L'CCHP_L4(,R3)          BUMP BUFFER BY STR LEN
  SPACE 1
CCHPEXIT DS 0H
  LM   R12,R14,WA1_CCHPSAVE
  XR   R15,R15
  BR   R14
  SPACE 1
*****
* CCHP - CONSTANTS
*****
CCHP_L1 DC  C'SET CHCAT=''
CCHP_L2 DC  C''';SET CHLOGID=''
CCHP_L3 DC  C''';SET CHSDESC=''
CCHP_L4 DC  C''';SET CHNEEDBY='12/31/10';
  LTORG
  DROP R12
  EJECT
  TITLE 'BC1PNM60 - ADD LOG VALUES FOR AN ELEMENT ACTION'
*****
* BC1PNM60 - CTLG
* AT THIS POINT, THE API BUFFER CONTAINS THE "OPLG" CMD FOLLOWED *
* BY THE "CLEAR" CMD. THE OLCAT, OLLOC AND OLTYPE "SET" COMMANDS *
* ARE ALREADY PLACED IN THE BUFFER.
*
* THIS ROUTINE WILL ADD "SET" CMD TO UPDATE THE TEXT PORTION OF *
* THE RECORD.
*
* REGISTER USAGE:
* R1-R4: WORK
* R8: EXIT CONTROL BLOCK ANCHOR
* R9: ADDR OF ENETM STRUCTURE (USER DEFINITION TABLE)
* R12: CTLGRUTN BASE ADDRESS
* R13: WA1DSECT STRUCTURE ADDRESS
*****

```

```

CTLGRUTN DS    0H
    STM  R12,R14,WA1_CTLGSAVE
    LR   R12,R15
    USING CTLGRUTN,R12
    L    R4,0(,R8)
    USING $ECBDS,R4
    L    R1,ECBFUNC
    SLL  R1,2
    B    CTLGTABL-4(R1)
    SPACE 1
CTLGTABL DS    0H
    B    CTLG0100          WHEN ECBFUNC = ECB$ADD
    B    CTLG0100          WHEN ECBFUNC = ECB$UPD
    B    CTLG0100          WHEN ECBFUNC = ECB$RETR
    B    CTLG0100          WHEN ECBFUNC = ECB$DEL
    B    CTLG0100          WHEN ECBFUNC = ECB$GEN
    B    CTLGEXIT          WHEN ECBFUNC = ECB$DISP
    B    CTLG0300          WHEN ECBFUNC = ECB$MOVE
    B    CTLG0200          WHEN ECBFUNC = ECB$ARCH
    B    CTLGEXIT          WHEN ECBFUNC = ECB$SUPV
    B    CTLGEXIT          WHEN ECBFUNC = ECB$PANL
    B    CTLGEXIT          WHEN ECBFUNC = ECB$CHGT
    B    CTLGEXIT          WHEN ECBFUNC = ECB$SINI
    B    CTLGEXIT          WHEN ECBFUNC = ECB$SINO
    B    CTLGEXIT          WHEN ECBFUNC = ECB$ENV
    B    CTLGEXIT          WHEN ECBFUNC = ECB$PRNT
    B    CTLG0300          WHEN ECBFUNC = ECB$TRNS
    B    CTLGEXIT          WHEN ECBFUNC = ECB$NUSE
    B    CTLG0100          WHEN ECBFUNC = ECB$REST
    DROP R4
    EJECT
*****
* FOR ELEMENT ACTIONS OF "ADD", "UPDATE", "GENERATE", "RETRIEVE" *
* "RESTORE" AND "DELETE".                                         *
* SET OLTEXT(1)='ACTION PROCESSING COMPLETED RC = ACTRC';
*      WHERE ACTRC = ACTION RETURN CODE VALUE.
* SET OLTEXT(2)='ELEMENT: ENV / SYS / SBS / TYPE / STA / ELE';
*      WHERE ENV, SYS, SBS, TYPE, STA AND ELE = ENDEVOR'S
*          INVENTORY LOCATION.
* SET OLTEXT(3)='ACTION: ELEACTION USER: USERID DATE: MM/DD/YY
*      TIME';
*      WHERE ELEACTION = THE ACTION VERB.
*      WHERE USERID = USER IDENTIFICATION.
*      WHERE DATE = CURRENT DATE.
*      WHERE TIME = CURRENT TIME.
* SET OLTEXT(4)='COMMENT: COMM';
*      WHERE COMM = ELEMENT ACTION'S COMMENT.
*****
    SPACE 1
*****
* COMPUTE TOTAL STRING LENGTH.                                     *
*****

```

CTLG0100	DS	0H	
LA	R2,L'CTLG_L1+8+L'CTLG_L2	LINE 1 LENGTH	
LA	R2,L'CTLG_L3+38+L'CTLG_L2(,R2)	ADD LINE 2 LENGTH	
LA	R2,L'CTLG_L4+8(,R2)	ADD 1ST PART OF LINE 3	
LA	R2,L'CTLG_L5+8(,R2)	ADD 2ND PART OF LINE 3	
LA	R2,L'CTLG_L6+8(,R2)	ADD 3RD PART OF LINE 3	
LA	R2,L'CTLG_L7+5+L'CTLG_L2(,R2)	ADD 4TH PART OF LINE 3	
LA	R2,L'CTLG_L8+L'REQCOMM+L'CTLG_L2(,R2)	ADD LINE 4 LENGTH	
LA	R2,255(,R2)	ADD MAX ELM LEN	
LA	R1,L'CTLG_L3+L'CTLG_L2	ADD MAX CONTROL TEXT FOR	
MH	R1,=H'5'	ELM NAME	
LA	R2,0(R1,R2)		
CH	R2,WA1_BUFF_LEN	IF BUFFER TOO SMALL FOR	
BH	CTLGEXIT	DATA, RETURN.	
SPACE 1			
* PLACE 1ST SET STRING INTO BUFFER.			
L	R4,0(,R8)	BUT 1ST ... CONVERT VARIABLE	
L	R4,ECBREQA-\$ECBDS(R4)	GET \$REQPDS C/B ADDRESS	
USING	\$REQPDS,R4		
ICM	R1,15,REQRTCOD	GET ACTION RETURN CODE	
DROP	R4		
CVD	R1,WA1_DBL	NOW CONVERT IT FROM	
OI	WA1_DBL+7,X'0F'	BINARY TO DISPLAY	
UNPK	WA1_TARGET1,WA1_DBL		
MVC	WA1_TARGET2,CTLG_BLANKS	GET RID OF LEADING	
LA	R1,WA1_TARGET1	CHARACTER ZEROES.	
LA	R2,L'WA1_TARGET1		
CTLG0110	DS	0H	
CLI	0(R1),C'0'	PUT RESULT OF RC	
BNE	CTLG0112	IN TARGET2, PAD W/	
LA	R1,1(,R1)	BLANKS.	
BCT	R2,CTLG0110		
MVI	WA1_TARGET2,C'0'		
BZ	CTLG0114		
CTLG0112	DS	0H	
BCTR	R2,0		
EX	R2,CTLG_MVC1		
B	CTLG0114		
CTLG_MVC1	DS	0H	
MVC	WA1_TARGET2(*-*),0(R1)		
CTLG0114	DS	0H	
L	R3,WA1_BUFF_ADDR	R3 -> BUFFER	
MVC	0(L'CTLG_L1,R3),CTLG_L1	MOVE STRING 1 INTO BUFFER	
LA	R3,L'CTLG_L1(,R3)	BUMP BUFFER BY STR LEN	
MVC	0(L'WA1_TARGET2,R3),WA1_TARGET2	MOVE IN VARIABLE	
LA	R3,L'WA1_TARGET2(,R3)	BUMP BUFFER BY VAR LEN	
MVC	0(L'CTLG_L2,R3),CTLG_L2	MOVE DEL STRING INTO BUF	
LA	R3,L'CTLG_L2(,R3)	BUMP BUFFER BY DEL LEN	
SPACE 1			

```

* PLACE 2ND SET STRING INTO BUFFER.
    MVC  0(L'CTLG_L3,R3),CTLG_L3      MOVE STRING 2 INTO BUFFER
    LA   R3,L'CTLG_L3(,R3)          BUMP BUFFER BY STR LEN
    L    R5,0(,R8)
    USING $ECBDS,R5                R5-> $ECBDS EXIT C/B
    L    R4,4(,R8)                  PRESET R4 TO GEN/RETR/DEL
    CLC  ECBFUNC,=A(ECB$GEN)       $ENVDS C/B.
    BE   CTLG0120                  IF FUNC = GEN/RETR/DEL
    CLC  ECBFUNC,=A(ECB$RETR)       THEN ALL SET
    BE   CTLG0120                  ELSE SWITCH TO OTHER LOCATION
    CLC  ECBFUNC,=A(ECB$DEL)
    BE   CTLG0120
    L    R4,ENVNEXT-$ENVDS(,R4)

CTLG0120 DS 0H
    USING $ENVDS,R4
    MVC  0(L'ENVENVM,R3),ENVENVM    MOVE IN ENV VAR INTO BUFFER
    MVI  L'ENVENVM(R3),C'/'        THEN THE SLASH
    LA   R3,L'ENVENVM+1(,R3)       BUMP BUFFER BY ENV '/' LEN
    MVC  0(L'ENVSYSSTM,R3),ENVSYSSTM  MOVE IN SYS VAR INTO BUFFER
    MVI  L'ENVSYSSTM(R3),C'/'      THEN THE SLASH
    LA   R3,L'ENVSYSSTM+1(,R3)     BUMP BUFFER BY SYS '/' LEN
    MVC  0(L'ENVSUBSY,R3),ENVSUBSY  MOVE IN SBS VAR INTO BUFFER
    MVI  L'ENVSUBSY(R3),C'/'       THEN THE SLASH
    LA   R3,L'ENVSUBSY+1(,R3)      BUMP BUFFER BY SBS '/' LEN
    MVC  0(L'ENVTYPE,R3),ENVTYPE    MOVE IN TYP VAR INTO BUFFER
    MVI  L'ENVTYPE(R3),C'/'        THEN THE SLASH
    LA   R3,L'ENVTYPE+1(,R3)       BUMP BUFFER BY TYP '/' LEN
    MVC  0(L'ENVSTGCD,R3),ENVSTGCD  MOVE IN STG VAR INTO BUF
    MVI  L'ENVSTGCD(R3),C'/'      THEN THE SLASH
    LA   R3,L'ENVSTGCD+1(,R3)     BUMP BUFFER BY STG '/' LEN
    LA   R6,2                      R6 = OLTEXT STRING NUMBER
    LH   R2,ENVEAOFF               R2 = ELEMENT AREA ADDRESS
    LA   R2,$ENVDS(R2)
    XR   R1,R1
    ICM  R1,3,0(R2)                R1 = ELEMENT NAME LENGTH
    LA   R2,2(,R2)                  R2 -> START OF ELMN NAME
    CH   R1,=H'18'                 IF NAME CAN FIT IN THIS STR
    BNH  CTLG0150                  THEN GO TO LAST MOVE.
    MVC  0(18,R3),0(R2)            PUT 1ST 18 CHARS IN BUF
    LA   R2,18(,R2)                BUMP SRC LOCATION BY 18
    LA   R3,18(,R3)                BUMP TGT BY 18 (NXT PLACE)
    SH   R1,=H'18'                 ADJUST LENGTH FIELD
    B    CTLG0140                  GO ADD DELIMITOR TO STRING
    SPACE 1

```

CTLG0130	DS	0H	
	LA	R6,1(,R6)	COMPUTE NEXT STRING NUMBER
	MVC	0(L'CTLG_L3,R3),CTLG_L3	MOVE STRING 2 AGAIN IN BUF
	LA	R14,OLTEXT_TBL1(R6)	GET CORRECT CHAR STRING NUM
	MVC	11(1,R3),0(R14)	OVERRIDE STRING NUM IN BUF
	MVC	17(8,R3),CTLG_BLANKS	CLEAR OUT 'ELEMENT:' LITR
	LA	R3,L'CTLG_L3(,R3)	BUMP BUF LOC BY STR SIZE
	CH	R1,=H'56'	IF NAME CAN ALL FIT W/IN THE
	BNH	CTLG0150	OLTEXT, GO TO LAST MOVE.
	MVC	0(56,R3),0(R2)	ELSE MOVE NEXT 56 CHARS
	LA	R2,56(,R2)	BUMP ELMN NAME POINTER
	LA	R3,56(,R3)	BUMP BUFFER POINTER
	SH	R1,=H'56'	DECREASE LENGTH LEFT SIZE
CTLG0140	DS	0H	
	MVC	0(L'CTLG_L2,R3),CTLG_L2	ADD STRING DEL TO BUFF
	LA	R3,L'CTLG_L2(,R3)	BUMP BUFFER BY DEL LENGTH
	B	CTLG0130	MOVE IN NEXT PART OF NAME
	SPACE	1	
CTLG0150	DS	0H	
	LTR	R1,R1	IF NOTHING LEFT
	BZ	CTLG0155	THEN GO PLACE DEL TEXT
	BCTR	R1,0	
	EX	R1,CTLG0156	MOVE IN REMAINING NAME
	LA	R3,1(R1,R3)	ADJUST BUFFER POINTER
CTLG0155	DS	0H	
	MVC	0(L'CTLG_L2,R3),CTLG_L2	ADD STRING DEL TO BUFF
	LA	R3,L'CTLG_L2(,R3)	BUMP BUFFER BY DEL LENGTH
	LA	R6,1(,R6)	BUMP STRING NUM FOR OLTEXT
	B	CTLG0180	GO PLACE 3RD STRING IN BUF
CTLG0156	MVC	0(*-*,R3),0(R2)	
	DROP	R4	
	SPACE	1	

```

* PLACE 3RD SET STRING INTO BUFFER.
CTLG0180 DS    0H
    MVC  0(L'CTLG_L4,R3),CTLG_L4      MOVE FIRST PART OF STR 3
    LA   R14,OLTEXT_TBL1(R6)        GET CORRECT CHAR STRING NUM
    MVC  11(1,R3),0(R14)          OVERRIDE STRING NUM IN BUF
    LA   R3,L'CTLG_L4(,R3)        BUMP BUFFER BY STR LEN
    LA   R6,1(,R6)                BUMP STR NUM FOR NEXT
    MVC  0(L'ECBFUNAM,R3),ECBFUNAM  MOVE IN ACTION VARIABLE
    LA   R3,L'ECBFUNAM(,R3)        BUMP BUFFER BY VAR LEN
    MVC  0(L'CTLG_L5,R3),CTLG_L5    MOVE IN 2ND PART OF STR 3
    LA   R3,L'CTLG_L5(,R3)        BUMP BUFFER BY STR LEN
    MVC  0(L'ECBUSER,R3),ECBUSER    MOVE IN USERID VARIABLE
    LA   R3,L'ECBUSER(,R3)        BUMP BUFFER BY VAR LEN
    MVC  0(L'CTLG_L6,R3),CTLG_L6    MOVE IN 3RD PART OF STR 3
    LA   R3,L'CTLG_L6(,R3)        BUMP BUFFER BY STR LEN
    MVC  0(L'WA1_CURR_DATE,R3),WA1_CURR_DATE  MOVE IN DATE VAR
    LA   R3,L'WA1_CURR_DATE(,R3)    BUMP BUFFER BY VAR LEN
    MVC  0(L'CTLG_L7,R3),CTLG_L7    MOVE IN 4TH PART OF STR 3
    LA   R3,L'CTLG_L7(,R3)        BUMP BUFFER BY STR LEN
    MVC  0(L'WA1_CURR_TIME,R3),WA1_CURR_TIME  MOVE IN TIME VAR
    LA   R3,L'WA1_CURR_TIME(,R3)    BUMP BUFFER BY VAR LEN
    MVC  0(L'CTLG_L2,R3),CTLG_L2    MOVE DEL STR INTO BUFFER
    LA   R3,L'CTLG_L2(,R3)        BUMP BUFFER BY DEL STR
    DROP R5
    SPACE 1

* PLACE 4TH SET STRING INTO BUFFER.
    MVC  0(L'CTLG_L8,R3),CTLG_L8      MOVE STRING 4 INTO BUFFER
    LA   R14,OLTEXT_TBL1(R6)        GET CORRECT CHAR STRING NUM
    MVC  11(1,R3),0(R14)          OVERRIDE STRING NUM IN BUF
    LA   R3,L'CTLG_L8(,R3)        BUMP BUFFER BY STR LEN
    L    R4,0(,R8)
    L    R4,ECBREQA-$ECBDS(,R4)    R4-> $REQPDS EXIT C/B
    USING $REQPDS,R4
    MVC  0(L'REQCOMM,R3),REQCOMM    MOVE IN COMMENT VAR
    LA   R3,L'REQCOMM(,R3)        BUMP BUFFER BY VAR LEN
    DROP R4
    MVC  0(L'CTLG_L2,R3),CTLG_L2    MOVE DEL STR INTO BUFFER
    LA   R3,L'CTLG_L2(,R3)        BUMP BUFFER BY DEL STR
    SPACE 1
    L    R14,WA1_BUFF_ADDR
    SR   R3,R14
    L    R14,WA1_DATALEN_ADDR
    STH  R3,0(R14)
    B    CTLGEXIT                  RETURN -- BUFFER UPDATED.
EJECT

```

```

*****
* FOR ELEMENT ACTIONS OF "ARCHIVE" *
*   SET OLTEXT(1)='ACTION PROCESSING COMPLETED RC = ACTRC'; *
*   WHERE ACTRC = ACTION RETURN CODE VALUE. *
*   SET OLTEXT(2)='ELEMENT: ENV / SYS / SBS / TYPE / STA / ELE'; *
*   WHERE ENV, SYS, SBS, TYPE, STA AND ELE = ENDEVOR'S *
*   INVENTORY LOCATION. *
*   SET OLTEXT(3)='TARGET: FI2DSN FI2MBR'; *
*   WHERE FI2DSN = TO DATA SET. *
*   WHERE FI2MBR = TO MEMBER. *
*   SET OLTEXT(4)='ACTION: ELECTION USER: USERID DATE: MM/DD/YY * *
*   TIME'; *
*   WHERE ELECTION = THE ACTION VERB. *
*   WHERE USERID = USER IDENTIFICATION. *
*   WHERE DATE = CURRENT DATE. *
*   WHERE TIME = CURRENT TIME. *
*   SET OLTEXT(5)='COMMENT: COMM'; *
*   WHERE COMM = ELEMENT ACTION'S COMMENT. *
*****
SPACE 1
*****
* COMPUTE TOTAL STRING LENGTH. *
*****
CTLG0200 DS 0H
LA R2,L'CTLG_L1+8+L'CTLG_L2      LINE 1 LENGTH
LA R2,L'CTLG_L3+38+L'CTLG_L2(,R2) ADD LINE 2 LENGTH
LA R2,L'CTLG_L9+L'FILDSN+1(,R2)  ADD 1ST PART OF LINE 3
LA R2,L'FIELD$MEM+L'CTLG_L2(,R2) ADD 2ND PART OF LINE 3
LA R2,L'CTLG_L10+8(,R2)          ADD 1ST PART OF LINE 4
LA R2,L'CTLG_L5+8(,R2)          ADD 2ND PART OF LINE 4
LA R2,L'CTLG_L6+8(,R2)          ADD 3RD PART OF LINE 4
LA R2,L'CTLG_L7+5+L'CTLG_L2(,R2) ADD 4TH PART OF LINE 4
LA R2,L'CTLG_L11+L'REQCOMM+L'CTLG_L2(,R2) ADD LINE 4 LEN
LA R2,255(,R2)                  ADD MAX ELM LEN
LA R1,L'CTLG_L3+L'CTLG_L2      ADD MAX CONTROL TEXT FOR
MH R1,=H'5'                      ELM NAME
LA R2,0(R1,R2)
CH R2,WA1_BUFF_LEN              IF BUFFER TOO SMALL FOR
BH CTLGEXIT                     DATA, RETURN.
SPACE 1
* PLACE 1ST SET STRING INTO BUFFER.
L  R4,0(,R8)                    BUT 1ST ... CONVERT VARIABLE
L  R4,ECBREQA-$ECBDS(,R4)
USING $REQPDS,R4
ICM R1,15,REQRTCOD            GET ACTION RETURN CODE
DROP R4
CVD R1,WA1_DBL
OI  WA1_DBL+7,X'0F'
UNPK WA1_TARGET1,WA1_DBL
MVC WA1_TARGET2,CTLG_BLANKS    GET RID OF LEADING
LA R1,WA1_TARGET1               CHARACTER ZEROES.
LA R2,L'WA1_TARGET1

```

CTLG0210	DS	0H	
	CLI	0(R1),C'0'	PUT RESULT OF RC
	BNE	CTLG0212	IN TARGET2, PAD W/
	LA	R1,1(,R1)	BLANKS.
	BCT	R2,CTLG0210	
	MVI	WA1_TARGET2,C'0'	
	BZ	CTLG0214	
CTLG0212	DS	0H	
	BCTR	R2,0	
	EX	R2,CTLG_MVC2	
	B	CTLG0214	
CTLG_MVC2	DS	0H	
	MVC	WA1_TARGET2(*-*),0(R1)	
CTLG0214	DS	0H	
	L	R3,WA1_BUFF_ADDR	R3 -> BUFFER
	MVC	0(L'CTLG_L1,R3),CTLG_L1	MOVE STRING 1 INTO BUFFER
	LA	R3,L'CTLG_L1(,R3)	BUMP BUFFER BY STR LEN
	MVC	0(L'WA1_TARGET2,R3),WA1_TARGET2	MOVE IN VARIABLE
	LA	R3,L'WA1_TARGET2(,R3)	BUMP BUFFER BY VAR LEN
	MVC	0(L'CTLG_L2,R3),CTLG_L2	MOVE DEL STRING INTO BUF
	LA	R3,L'CTLG_L2(,R3)	BUMP BUFFER BY DEL LEN
	SPACE 1		
*	PLACE 2ND SET STRING INTO BUFFER.		
	MVC	0(L'CTLG_L3,R3),CTLG_L3	MOVE STRING 2 INTO BUFFER
	LA	R3,L'CTLG_L3(,R3)	BUMP BUFFER BY STR LEN
	L	R5,0(,R8)	R5-> \$ECBDS EXIT C/B
	USING	\$ECBDS,R5	
	L	R4,4(,R8)	R4-> \$ENVDS EXIT C/B FOR ARCHIVE
*	USING	\$ENVDS,R4	
	MVC	0(L'ENVENVM,R3),ENVENVM	MOVE IN ENV VAR INTO BUFFER
	MVI	L'ENVENVM(R3),C'/'	THEN THE SLASH
	LA	R3,L'ENVENVM+1(,R3)	BUMP BUFFER BY ENV '/' LEN
	MVC	0(L'ENVSYSYM,R3),ENVSYSYM	MOVE IN SYS VAR INTO BUFFER
	MVI	L'ENVSYSYM(R3),C'/'	THEN THE SLASH
	LA	R3,L'ENVSYSYM+1(,R3)	BUMP BUFFER BY SYS '/' LEN
	MVC	0(L'ENVSUBSY,R3),ENVSUBSY	MOVE IN SBS VAR INTO BUFFER
	MVI	L'ENVSUBSY(R3),C'/'	THEN THE SLASH
	LA	R3,L'ENVSUBSY+1(,R3)	BUMP BUFFER BY SYS '/' LEN
	MVC	0(L'ENVTYPE,R3),ENVTYPE	MOVE IN TYP VAR INTO BUFFER
	MVI	L'ENVTYPE(R3),C'/'	THEN THE SLASH

LA	R3,L'ENVTYPE+1(,R3)	BUMP BUFFER BY TYP '/' LEN
MVC	0(L'ENVSTGCD,R3),ENVSTGCD	MOVE IN STG VAR INTO BUF
MVI	L'ENVSTGCD(R3),C'/'	THEN THE SLASH
LA	R3,L'ENVSTGCD+1(,R3)	BUMP BUFFER BY STG '/' LEN
LA	R6,2	R6 = OLTEXT STRING NUMBER
LH	R2,ENVEAOFF	R2 = ELEMENT AREA ADDRESS
LA	R2,\$ENVDS(R2)	
XR	R1,R1	
ICM	R1,3,0(R2)	R1 = ELEMENT NAME LENGTH
LA	R2,2(,R2)	R2 -> START OF ELMN NAME
CH	R1,=H'18'	IF NAME CAN FIT IN THIS STR
BNH	CTLG0250	THEN GO TO LAST MOVE.
MVC	0(18,R3),0(R2)	PUT 1ST 18 CHARS IN BUF
LA	R2,18(,R2)	BUMP SRC LOCATION BY 18
LA	R3,18(,R3)	BUMP TGT BY 18 (NXT PLACE)
SH	R1,=H'18'	ADJUST LENGTH FIELD
B	CTLG0240	GO ADD DELIMITOR TO STRING
SPACE	1	
CTLG0230	DS 0H	
LA	R6,1(,R6)	COMPUTE NEXT STRING NUMBER
MVC	0(L'CTLG_L3,R3),CTLG_L3	MOVE STRING 2 AGAIN IN BUF
LA	R14,OLTEXT_TBL1(R6)	GET CORRECT CHAR STRING NUM
MVC	11(1,R3),0(R14)	OVERRIDE STRING NUM IN BUF
MVC	17(8,R3),CTLG_BLANKS	CLEAR OUT 'ELEMENT:' LITR
LA	R3,L'CTLG_L3(,R3)	BUMP BUF LOC BY STR SIZE
CH	R1,=H'56'	IF THIS CAN ALL FIT W/IN THE
BNH	CTLG0250	OLTEXT, GO TO LAST MOVE.
MVC	0(56,R3),0(R2)	ELSE MOVE NEXT 56 CHARS
LA	R2,56(,R2)	BUMP ELMN NAME POINTER
LA	R3,56(,R3)	BUMP BUF POINTER
SH	R1,=H'56'	DECREASE LENGTH LEFT SIZE
CTLG0240	DS 0H	
MVC	0(L'CTLG_L2,R3),CTLG_L2	ADD STRING DEL TO BUFF
LA	R3,L'CTLG_L2(,R3)	BUMP BUFFER BY DEL LENGTH
B	CTLG0230	MOVE IN NEXT PART OF NAME
SPACE	1	
CTLG0250	DS 0H	
LTR	R1,R1	IF NOTHING LEFT
BZ	CTLG0255	THEN GO PLACE DEL TEXT
BCTR	R1,0	
EX	R1,CTLG0256	MOVE IN REMAINING NAME
LA	R3,1(R1,R3)	ADJUST BUFFER POINTER
CTLG0255	DS 0H	
MVC	0(L'CTLG_L2,R3),CTLG_L2	ADD STRING DEL TO BUFF
LA	R3,L'CTLG_L2(,R3)	BUMP BUFFER BY DEL LENGTH
LA	R6,1(,R6)	BUMP STRING NUM FOR OLTEXT
B	CTLG0280	GO PLACE 3RD STRING
CTLG0256	MVC 0(*-*,R3),0(R2)	
DROP	R4	
SPACE	1	

```

* PLACE 3RD SET STRING INTO BUFFER.
CTLG0255 DS    0H
CTLG0280 DS    0H
        L    R4,4(,R8)
        L    R4,ENVNEXT-$ENVDS(,R4)      R4-> $FILDS FOR ARCHIVE
        L    R4,ENVFCBA-$ENVDS(,R4)      MOVE STRING 3 INTO BUFFER
        MVC 0(L'CTLG_L9A,R3),CTLG_L9A   FOR ARCHIVE.
        LA   R14,OLTEXT_TBL1(R6)        GET CORRECT CHAR STRING NUM
        MVC 11(1,R3),0(R14)            OVERRIDE STRING NUM IN BUF
        USING $FILDS,R4
        LA   R3,L'CTLG_L9(,R3)        BUMP BUFFER BY STR LEN
        LA   R6,1(,R6)                ADD 1 TO STR COUNTER
        MVC 0(L'FILDSN,R3),FILDSN    MOVE DSN VAR INTO BUF
        MVI L'FILDSN(R3),C' '
        LA   R3,L'FILDSN+1(,R3)       MOVE IN AN EXTRA BLANK
        MVC 0(L'FIELDSMEM,R3),FIELDSMEM  BUMP BUFF BY DSN ' ' LEN
        LA   R3,L'FIELDSMEM(,R3)      MOVE MEM VAR INTO BUF
        MVC 0(L'CTLG_L2,R3),CTLG_L2   BUMP BUFF BY MEM LEN
        LA   R3,L'CTLG_L2(,R3)        MOVE DEL STR INTO BUFFER
        LA   R3,L'CTLG_L2(,R3)        BUMP BUFFER BY DEL LEN
        DROP R4,R5
        SPACE 1
* PLACE 4TH SET STRING INTO BUFFER.
        MVC 0(L'CTLG_L10,R3),CTLG_L10  MOVE FIRST PART OF STR 4
        LA   R14,OLTEXT_TBL1(R6)      GET CORRECT CHAR STRING NUM
        MVC 11(1,R3),0(R14)          OVERRIDE STRING NUM IN BUF
        LA   R3,L'CTLG_L10(,R3)       BUMP BUFFER BY STR LEN
        LA   R6,1(,R6)                ADD 1 TO STR COUNTER
        L    R4,0(,R8)                R4-> $ECBDS EXIT C/B
        USING $ECBDS,R4
        MVC 0(L'ECBFUNAM,R3),ECBFUNAM  MOVE IN ACTION VARIABLE
        LA   R3,L'ECBFUNAM(,R3)      BUMP BUFFER BY VAR LEN
        MVC 0(L'CTLG_L5,R3),CTLG_L5   MOVE IN 2ND PART OF STR 4
        LA   R3,L'CTLG_L5(,R3)        BUMP BUFFER BY STR LEN
        MVC 0(L'ECBUSER,R3),ECBUSER  MOVE IN USERID VARIABLE
        LA   R3,L'ECBUSER(,R3)        BUMP BUFFER BY VAR LEN
        MVC 0(L'CTLG_L6,R3),CTLG_L6   MOVE IN 3RD PART OF STR 4
        LA   R3,L'CTLG_L6(,R3)        BUMP BUFFER BY STR LEN
        MVC 0(L'WA1_CURR_DATE,R3),WA1_CURR_DATE  MOVE IN DATE VAR
        LA   R3,L'WA1_CURR_DATE(,R3)  BUMP BUFFER BY VAR LEN
        MVC 0(L'CTLG_L7,R3),CTLG_L7   MOVE IN 4TH PART OF STR 4
        LA   R3,L'CTLG_L7(,R3)        BUMP BUFFER BY STR LEN
        MVC 0(L'WA1_CURR_TIME,R3),WA1_CURR_TIME  MOVE IN TIME VAR
        LA   R3,L'WA1_CURR_TIME(,R3)  BUMP BUFFER BY VAR LEN
        MVC 0(L'CTLG_L2,R3),CTLG_L2   MOVE DEL STR INTO BUFFER
        LA   R3,L'CTLG_L2(,R3)        BUMP BUFFER BY DEL STR
        DROP R4
        SPACE 1

```

```

* PLACE 5TH SET STRING INTO BUFFER.
    CH   R6,=H'10'
    BNL  CTLG0290
    MVC  0(L'CTLG_L11,R3),CTLG_L11 MOVE STRING 5 INTO BUFFER
    LA   R14,OLTEXT_TBL1(R6) GET CORRECT CHAR STRING NUM
    MVC  11(1,R3),0(R14) OVERRIDE STRING NUM IN BUF
    LA   R3,L'CTLG_L11(,R3) BUMP BUFFER BY STR LEN
    B    CTLG0295
CTLG0290 DS  0H NOTE: 10 MAXIMUM OLTEXT SUBR
    MVC  0(L'CTLG_L14,R3),CTLG_L14 MOVE STRING 5 INTO BUFFER
    LA   R3,L'CTLG_L14(,R3) BUMP BUFFER BY STR LEN
CTLG0295 DS  0H
    L   R4,0(,R8)
    L   R4,ECBREQA-$ECBDS(,R4) R4-> $REQPDS EXIT C/B
    USING $REQPDS,R4 MOVE IN COMMENT VAR
    MVC  0(L'REQCOMM,R3),REQCOMM BUMP BUFFER BY VAR LEN
    LA   R3,L'REQCOMM(,R3)
    DROP R4
    MVC  0(L'CTLG_L2,R3),CTLG_L2 MOVE DEL STR INTO BUFFER
    LA   R3,L'CTLG_L2(,R3) BUMP BUFFER BY DEL STR
    SPACE 1
    L   R14,WA1_BUFF_ADDR
    SR  R3,R14
    L   R14,WA1_DATALEN_ADDR
    STH R3,0(R14)
    B    CTLGEXIT RETURN -- BUFFER UPDATED.
    EJECT

*****
* FOR ELEMENT MOVE AND TRANSFER. *
* SET OLTEXT(1)='ACTION PROCESSING COMPLETED RC = ACTRC'; *
* WHERE ACTRC = ACTION RETURN CODE VALUE. *
* SET OLTEXT(2)='SOURCE ELEMENT: ENV / SYS / SBS / TYPE / STA / *
* ELE'; *
* WHERE ENV, SYS, SBS, TYPE, STA AND ELE = ENDEVOR'S *
* FROM INVENTORY LOCATION. *
* SET OLTEXT(3)='TARGET ELEMENT: ENV / SYS / SBS / TYPE / STA / *
* ELE'; *
* WHERE ENV, SYS, SBS, TYPE, STA AND ELE = ENDEVOR'S *
* TO INVENTORY LOCATION. *
* SET OLTEXT(4)='ACTION: ELECTION USER: USERID DATE: MM/DD/YY *
* TIME'; *
* WHERE ELECTION = THE ACTION VERB. *
* WHERE USERID = USER IDENTIFICATION. *
* WHERE DATE = CURRENT DATE. *
* WHERE TIME = CURRENT TIME. *
* SET OLTEXT(5)='COMMENT: COMM'; *
* WHERE COMM = ELEMENT ACTION'S COMMENT. *
*****
    SPACE 1

```

```
*****
* COMPUTE TOTAL STRING LENGTH. *
*****
CTLG0300 DS   0H
    LA R2,L'CTLG_L1+8+L'CTLG_L2           LINE 1 LENGTH
    LA R2,L'CTLG_L12+38+L'CTLG_L2(,R2)   ADD LINE 2 LENGTH
    LA R2,L'CTLG_L13+38+L'CTLG_L2(,R2)   ADD LINE 3 LENGTH
    LA R2,L'CTLG_L10+8(,R2)              ADD 1ST PART OF LINE 4
    LA R2,L'CTLG_L5+8(,R2)              ADD 2ND PART OF LINE 4
    LA R2,L'CTLG_L6+8(,R2)              ADD 3RD PART OF LINE 4
    LA R2,L'CTLG_L7+5+L'CTLG_L2(,R2)   ADD 4TH PART OF LINE 4
    LA R2,L'CTLG_L11+L'REQCOMM+L'CTLG_L2(,R2) ADD LINE 4 LEN
    LA R2,512(,R2)                      ADD MAX ELM LEN
    LA R1,L'CTLG_L3+L'CTLG_L2           ADD MAX CONTROL TEXT FOR
    MH R1,=H'10'                        ELM NAME
    LA R2,0(R1,R2)
    CH R2,WA1_BUFF_LEN                 IF BUFFER TOO SMALL FOR
    BH CTLGEXIT                       DATA, RETURN.

    SPACE 1

* PLACE 1ST SET STRING INTO BUFFER.
    L  R4,0(,R8)                         BUT 1ST ... CONVERT VARIABLE
    L  R4,ECBREQA-$ECBDS(,R4)
    USING $REQPDS,R4
    ICM R1,15,REQRTCOD                GET ACTION RETURN CODE
    DROP R4
    CVD R1,WA1_DBL                     NOW CONVERT IT FROM
    OI  WA1_DBL+7,X'0F'                 BINARY TO DISPLAY
    UNPK WA1_TARGET1,WA1_DBL
    MVC WA1_TARGET2,CTLG_BLANKS        GET RID OF LEADING
    LA R1,WA1_TARGET1                  CHARACTER ZEROES.
    LA R2,L'WA1_TARGET1

CTLG0310 DS   0H
    CLI 0(R1),C'0'                      PUT RESULT OF RC
    BNE CTLG0312
    LA R1,1(,R1)
    BCT R2,CTLG0310
    MVI WA1_TARGET2,C'0'
    BZ  CTLG0314

CTLG0312 DS   0H
    BCTR R2,0
    EX  R2,CTLG_MVC3
    B   CTLG0314

CTLG_MVC3 DS   0H
    MVC WA1_TARGET2(*--),0(R1)

CTLG0314 DS   0H
    L   R3,WA1_BUFF_ADDR               R3 -> BUFFER
    MVC 0(L'CTLG_L1,R3),CTLG_L1      MOVE STRING 1 INTO BUFFER
    LA  R3,L'CTLG_L1(,R3)             BUMP BUFFER BY STR LEN
    MVC 0(L'WA1_TARGET2,R3),WA1_TARGET2 MOVE IN VARIABLE
    LA  R3,L'WA1_TARGET2(,R3)         BUMP BUFFER BY VAR LEN
    MVC 0(L'CTLG_L2,R3),CTLG_L2      MOVE DEL STRING INTO BUF
    LA  R3,L'CTLG_L2(,R3)             BUMP BUFFER BY DEL LEN

    SPACE 1
```

```

* PLACE 2ND SET STRING INTO BUFFER.
MVC 0(L'CTLG_L12,R3),CTLG_L12 MOVE STRING 2 INTO BUFFER
LA R3,L'CTLG_L12(,R3) BUMP BUFFER BY STR LEN
L R4,4(,R8) R4-> $ENVDS EXIT C/B
USING $ENVDS,R4 FOR SOURCE SIDE.
MVC 0(L'ENVENVM,R3),ENVENVM MOVE IN ENV VAR INTO BUFFER
MVI L'ENVENVM(R3),C'/' THEN THE SLASH
LA R3,L'ENVENVM+1(,R3) BUMP BUFFER BY ENV '/' LEN
MVC 0(L'ENVSYSTM,R3),ENVSYSTM MOVE IN SYS VAR INTO BUFFER
MVI L'ENVSYSTM(R3),C'/' THEN THE SLASH
LA R3,L'ENVSYSTM+1(,R3) BUMP BUFFER BY SYS '/' LEN
MVC 0(L'ENVSUBSY,R3),ENVSUBSY MOVE IN SBS VAR INTO BUFFER
MVI L'ENVSUBSY(R3),C'/' THEN THE SLASH
LA R3,L'ENVSUBSY+1(,R3) BUMP BUFFER BY SYS '/' LEN
MVC 0(L'ENVTYPE,R3),ENVTYPE MOVE IN TYP VAR INTO BUFFER
MVI L'ENVTYPE(R3),C'/' THEN THE SLASH
LA R3,L'ENVTYPE+1(,R3) BUMP BUFFER BY TYP '/' LEN
MVC 0(L'ENVSTGCD,R3),ENVSTGCD MOVE IN STG VAR INTO BUF
MVI L'ENVSTGCD(R3),C'/' THEN THE SLASH
LA R3,L'ENVSTGCD+1(,R3) BUMP BUFFER BY STG '/' LEN
LA R6,2 R6 = OLTEXT STRING NUMBER
LH R2,ENVEAOFF
LA R2,$ENVDS(R2)
XR R1,R1
ICM R1,3,0(R2) R1 = ELEMENT NAME LENGTH
LA R2,2(,R2) R2 -> START OF ELMN NAME
CH R1,=H'11' IF NAME CAN FIT IN THIS STR
BNH CTLG0330 THEN GO TO LAST MOVE.
MVC 0(11,R3),0(R2) PUT 1ST 11 CHARS IN BUF
LA R2,11(,R2) BUMP SRC LOCATION BY 11
LA R3,11(,R3) BUMP TGT BY 11 (NXT PLACE)
SH R1,=H'11' ADJUST LENGTH FIELD
B CTLG0325 GO ADD DELIMITOR TO STRING
SPACE 1
CTLG0320 DS 0H
LA R6,1(,R6) COMPUTE NEXT STRING NUMBER
MVC 0(L'CTLG_L12,R3),CTLG_L12 MOVE STRING 2 AGAIN IN BUF
LA R14,OLTEXT_TBL1(R6) GET CORRECT CHAR STRING NUM
MVC 11(1,R3),0(R14) OVERRIDE STRING NUM IN BUF
MVC 17(15,R3),CTLG_BLANKS BLANK OUT LITERIALS IN L12
LA R3,L'CTLG_L12(,R3) BUMP BUF LOC BY STR SIZE
CH R1,=H'49' IF THIS CAN ALL FIT W/IN THE
BNH CTLG0330 OLTEXT, GO TO LAST MOVE.
MVC 0(49,R3),0(R2) ELSE MOVE NEXT 49 CHARS
LA R2,49(,R2) BUMP ELEMENT NAME POINTER
LA R3,49(,R3) BUMP BUFFER POINTER
SH R1,=H'49' DECREASE LENGTH LEFT SIZE
CTLG0325 DS 0H
MVC 0(L'CTLG_L2,R3),CTLG_L2 ADD STRING DEL TO BUFF
LA R3,L'CTLG_L2(,R3) BUMP BUFFER BY DEL LENGTH
B CTLG0320 MOVE IN NEXT PART OF NAME
SPACE 1

```

CTLG0330	DS	0H	
	LTR	R1,R1	IF NOTHING LEFT
	BZ	CTLG0335	THEN GO PLACE DEL TEXT
	BCTR	R1,0	
	EX	R1,CTLG0336	MOVE IN REMAINING NAME
	LA	R3,1(R1,R3)	ADJUST BUFFER POINTER
CTLG0335	DS	0H	
	MVC	0(L'CTLG_L2,R3),CTLG_L2	ADD STRING DEL TO BUFF
	LA	R3,L'CTLG_L2(,R3)	BUMP BUFFER BY DEL LENGTH
	LA	R6,1(,R6)	BUMP STRING NUM FOR OLTEXT
	B	CTLG0340	GO PLACE 3RD STRING IN BUF
CTLG0336	MVC	0(*--,R3),0(R2)	
	DROP	R4	
	SPACE	1	
* PLACE 3RD SET STRING INTO BUFFER.			
CTLG0340	DS	0H	
	MVC	0(L'CTLG_L13,R3),CTLG_L13	MOVE STRING 3 INTO BUFFER
	LA	R14,OLTEXT_TBL1(R6)	GET CORRECT CHAR STRING NUM
	MVC	11(1,R3),0(R14)	OVERRIDE STRING NUM IN BUF
	LA	R3,L'CTLG_L13(,R3)	BUMP BUFFER BY STR LEN
	L	R4,4(,R8)	
	L	R4,ENVNEXT-\$ENVDS(,R4)	R4-> \$ENVDS EXIT C/B
	USING	\$ENVDS,R4	FOR TARGET SIDE.
	MVC	0(L'ENVENVM,R3),ENVENVM	MOVE IN ENV VAR INTO BUFFER
	MVI	L'ENVENVM(R3),C'/'	THEN THE SLASH
	LA	R3,L'ENVENVM+1(,R3)	BUMP BUFFER BY ENV '/' LEN
	MVC	0(L'ENVSYSYSTM,R3),ENVSYSYSTM	MOVE IN SYS VAR INTO BUFFER
	MVI	L'ENVSYSYSTM(R3),C'/'	THEN THE SLASH
	LA	R3,L'ENVSYSYSTM+1(,R3)	BUMP BUFFER BY SYS '/' LEN
	MVC	0(L'ENVSUBSY,R3),ENVSUBSY	MOVE IN SBS VAR INTO BUFFER
	MVI	L'ENVSUBSY(R3),C'/'	THEN THE SLASH
	LA	R3,L'ENVSUBSY+1(,R3)	BUMP BUFFER BY SBS '/' LEN
	MVC	0(L'ENVTYPE,R3),ENVTYPE	MOVE IN TYP VAR INTO BUFFER
	MVI	L'ENVTYPE(R3),C'/'	THEN THE SLASH
	LA	R3,L'ENVTYPE+1(,R3)	BUMP BUFFER BY TYP '/' LEN
	MVC	0(L'ENVSTGCD,R3),ENVSTGCD	MOVE IN STG VAR INTO BUF
	MVI	L'ENVSTGCD(R3),C'/'	THEN THE SLASH
	LA	R3,L'ENVSTGCD+1(,R3)	BUMP BUFFER BY STG '/' LEN
	LH	R2,ENVEAOFF	
	LA	R2,\$ENVDS(R2)	R2 = ELEMENT AREA ADDRESS
	XR	R1,R1	
	ICM	R1,3,0(R2)	R1 = ELEMENT NAME LENGTH
	LA	R2,2(,R2)	R2 -> START OF ELMN NAME
	CH	R1,=H'11'	IF NAME CAN FIT IN THIS STR
	BNH	CTLG0360	THEN GO TO LAST MOVE.
	MVC	0(11,R3),0(R2)	PUT 1ST 11 CHARS IN BUF
	LA	R2,11(,R2)	BUMP SRC LOCATION BY 11
	LA	R3,11(,R3)	BUMP TGT BY 11 (NXT PLACE)
	SH	R1,=H'11'	ADJUST LENGTH FIELD
	B	CTLG0355	GO ADD DELIMITOR TO STRING
	SPACE	1	

CTLG0350	DS	0H	
	LA	R6,1(,R6)	COMPUTE NEXT STRING NUMBER
	CH	R6,=H'10'	IF STRING NUM BELOW 10
	BL	CTLG0352	USE SINGLE SUBR OLTEXT
	MVC	0(L'CTLG_L13A,R3),CTLG_L13A	
	LR	R15,R6	MOVE IN TEXT STRING W/ DBL
	SH	R15,=H'10'	OLTEXT SUBR NUMBER
	SLL	R15,1	
	LA	R14,OLTEXT_TBL2(R15)	GET 2 DIGIT OVERRIDE NUMBER
	MVC	11(2,R3),0(R14)	OVERRIDE OLTEXT NUMBER
	MVC	18(15,R3),CTLG_BLANKS	BLANK OUT LITERIALS
	LA	R3,L'CTLG_L13A(,R3)	BUMP BUF LOC BY STR SIZE
	B	CTLG0353	GO MOVE IN NAME
CTLG0352	DS	0H	
	MVC	0(L'CTLG_L13,R3),CTLG_L13	MOVE STRING 2 AGAIN IN BUF
	LA	R14,OLTEXT_TBL1(R6)	GET CORRECT CHAR STRING NUM
	MVC	11(1,R3),0(R14)	OVERRIDE STRING NUM IN BUF
	MVC	17(15,R3),CTLG_BLANKS	BLANK OUT LITERIALS
	LA	R3,L'CTLG_L13(,R3)	BUMP BUF LOC BY STR SIZE
CTLG0353	DS	0H	
	CH	R1,=H'49'	IF THIS CAN ALL FIT W/IN THE
	BNH	CTLG0360	OLTEXT, GO TO LAST MOVE.
	MVC	0(49,R3),0(R2)	ELSE MOVE NEXT 49 CHARS
	CH	R6,=H'10'	DO NOT GO OVER 10 LOG
	BL	CTLG0354	LINES WHILE PLACEING IN THE
	MVC	45(4,R3),=C' ... '	ELEMENT NAME. OLTEXT SUBR'S
	LA	R3,49(,R3)	MAX AT 12 LINES.
	B	CTLG0365	GO PLACE DEL STRING IN BUF
CTLG0354	DS	0H	
	LA	R2,49(,R2)	BUMP ELEMENT NAME POINTER
	LA	R3,49(,R3)	BUMP BUFFER POINTER
	SH	R1,=H'49'	DECREASE LENGTH LEFT SIZE
CTLG0355	DS	0H	
	MVC	0(L'CTLG_L2,R3),CTLG_L2	ADD STRING DEL TO BUFF
	LA	R3,L'CTLG_L2(,R3)	BUMP BUFFER BY DEL LENGTH
	B	CTLG0350	MOVE IN NEXT PART OF NAME
	SPACE	1	
CTLG0360	DS	0H	
	LTR	R1,R1	IF NOTHING LEFT
	BZ	CTLG0365	THEN GO PLACE DEL TEXT
	BCTR	R1,0	
	EX	R1,CTLG0366	MOVE IN REMAINING NAME
	LA	R3,1(R1,R3)	ADJUST BUFFER POINTER
CTLG0365	DS	0H	
	MVC	0(L'CTLG_L2,R3),CTLG_L2	ADD STRING DEL TO BUFF
	LA	R3,L'CTLG_L2(,R3)	BUMP BUFFER BY DEL LENGTH
	LA	R6,1(,R6)	BUMP STRING NUM FOR OLTEXT
	B	CTLG0370	GO PLACE 4TH STRING
CTLG0366	MVC	0(*-*-,R3),0(R2)	
	DROP	R4	
	SPACE	1	

```

* PLACE 4TH SET STRING INTO BUFFER.
CTLG0370 DS   0H
    CH R6,=H'10'           IF STRING NUM SINGLE DIGIT
    BL CTLG0372             GO USE SIGNAL DIGIT STR TEXT
    MVC 0(L'CTLG_L10A,R3),CTLG_L10A
    LR R15,R6               ELSE USE DBL DIGIT STR TEXT
    SH R15,=H'10'
    SLL R15,1
    LA R14,OLTEXT_TBL2(R15) GET DBL DIGIT NUMBER FROM TBL
    MVC 11(2,R3),0(R14)     OVERRIDE TEXT W/ DBL DIGIT
    LA R3,L'CTLG_L10A(,R3) BUMP BUFFER POINT BY STR TEXT
    B  CTLG0374             GO MOVE IN DATA

CTLG0372 DS   0H
    MVC 0(L'CTLG_L10,R3),CTLG_L10 MOVE FIRST PART OF STR 4
    LA R14,OLTEXT_TBL1(R6) GET CORRECT CHAR STRING NUM
    MVC 11(1,R3),0(R14)     OVERRIDE STRING NUM IN BUF
    LA R3,L'CTLG_L10(,R3)  BUMP BUFFER BY STR LEN

CTLG0374 DS   0H
    LA R6,1(,R6)
    L  R4,0(,R8)           R4-> $ECBDS EXIT C/B
    USING $ECBDS,R4
    MVC 0(L'ECBFUNAM,R3),ECBFUNAM MOVE IN ACTION VARIABLE
    LA R3,L'ECBFUNAM(,R3)  BUMP BUFFER BY VAR LEN
    MVC 0(L'CTLG_L5,R3),CTLG_L5 MOVE IN 2ND PART OF STR 4
    LA R3,L'CTLG_L5(,R3)   BUMP BUFFER BY STR LEN
    MVC 0(L'ECBUSER,R3),ECBUSER MOVE IN USERID VARIABLE
    LA R3,L'ECBUSER(,R3)   BUMP BUFFER BY VAR LEN
    MVC 0(L'CTLG_L6,R3),CTLG_L6 MOVE IN 3RD PART OF STR 4
    LA R3,L'CTLG_L6(,R3)   BUMP BUFFER BY STR LEN
    MVC 0(L'WA1_CURR_DATE,R3),WA1_CURR_DATE MOVE IN DATE VAR
    LA R3,L'WA1_CURR_DATE(,R3) BUMP BUFFER BY VAR LEN
    MVC 0(L'CTLG_L7,R3),CTLG_L7 MOVE IN 4TH PART OF STR 4
    LA R3,L'CTLG_L7(,R3)   BUMP BUFFER BY STR LEN
    MVC 0(L'WA1_CURR_TIME,R3),WA1_CURR_TIME MOVE IN TIME VAR
    LA R3,L'WA1_CURR_TIME(,R3) BUMP BUFFER BY VAR LEN
    MVC 0(L'CTLG_L2,R3),CTLG_L2 MOVE DEL STR INTO BUFFER
    LA R3,L'CTLG_L2(,R3)   BUMP BUFFER BY DEL STR
    DROP R4
    SPACE 1

* PLACE 5TH SET STRING INTO BUFFER.
    CH R6,=H'10'           IF STRING NUM SINGLE DIGIT
    BL CTLG0382             GO USE SIGNAL DIGIT STR TEXT
    MVC 0(L'CTLG_L11A,R3),CTLG_L11A
    LR R15,R6               ELSE USE DBL DIGIT STR TEXT
    SH R15,=H'10'
    SLL R15,1
    LA R14,OLTEXT_TBL2(R15) GET DBL DIGIT NUMBER FROM TBL
    MVC 11(2,R3),0(R14)     OVERRIDE TEXT W/ DBL TEXT
    LA R3,L'CTLG_L11A(,R3) BUMP BUFFER PTR BY STR TEXT
    B  CTLG0384             GO MOVE DATA

```

```

CTLG0382 DS   0H
             MVC 0(L'CTLG_L11,R3),CTLG_L11 MOVE FIRST PART OF STR 4
             LA  R14,OLTEXT_TBL1(R6) GET CORRECT CHAR STRING NUM
             MVC 11(1,R3),0(R14) OVERRIDE STRING NUM IN BUF
             LA  R3,L'CTLG_L11(,R3) BUMP BUFFER BY STR LEN
CTLG0384 DS   0H
             LA  R6,1(,R6)
             L   R4,0(,R8)
             L   R4,ECBREQA-$ECBDS(,R4)
             USING $REQPDS,R4 R4-> $REQPDS EXIT C/B
             MVC 0(L'REQCOMM,R3),REQCOMM MOVE IN COMMENT VAR
             LA  R3,L'REQCOMM(,R3) BUMP BUFFER BY VAR LEN
             DROP R4
             MVC 0(L'CTLG_L2,R3),CTLG_L2 MOVE DEL STR INTO BUFFER
             LA  R3,L'CTLG_L2(,R3) BUMP BUFFER BY DEL STR
             SPACE 1
             L   R14,WA1_BUFF_ADDR
             SR  R3,R14
             L   R14,WA1_DATALEN_ADDR
             STH  R3,0(R14)
             B   CTLGEXIT
             SPACE 1
             RETURN -- BUFFER UPDATED.

CTLGEXIT DS   0H
             LM  R12,R14,WA1_CTLGSAVE
             XR  R15,R15
             BR  R14
             SPACE 1
CTLG_L1  DC   C'SET OLTEXT(1)='ACTION PROCESSING COMPLETED RC = '
CTLG_L2  DC   C''';'
CTLG_L3  DC   C'SET OLTEXT(2) = ''ELEMENT: '
CTLG_L4  DC   C'SET OLTEXT(3) = ''ACTION: '
CTLG_L5  DC   C' USER: '
CTLG_L6  DC   C' DATE: '
CTLG_L7  DC   C' TIME: '

```

```

CTLG_L8 DC    C'SET OLTEXT(4) =''COMMENT: '
CTLG_L9 DC    C'SET OLTEXT(3)=''SOURCE: '
CTLG_L9A DC   C'SET OLTEXT(3)=''TARGET: '
CTLG_L10 DC   C'SET OLTEXT(4) =''ACTION: '
CTLG_L10A DC  C'SET OLTEXT(XX) =''ACTION: '
CTLG_L11 DC   C'SET OLTEXT(5) =''COMMENT: '
CTLG_L11A DC  C'SET OLTEXT(XX) =''COMMENT: '
CTLG_L12 DC   C'SET OLTEXT(2) = ''SOURCE ELEMENT: '
CTLG_L13 DC   C'SET OLTEXT(3) = ''TARGET ELEMENT: '
CTLG_L13A DC  C'SET OLTEXT(XX) = ''TARGET ELEMENT: '
CTLG_L14 DC   C'SET OLTEXT(10) =''COMMENT: '
          SPACE 1
CTLG_BLANKS DC CL80'
          SPACE 1
OLTEXT_TBL1 DC C'0123456789'
OLTEXT_TBL2 DC C'1011121314151617181920'
          SPACE 1
LTORG
DROP R12
EJECT
TITLE 'BC1PNM60 - ADD LOG VALUES FOR A PACKAGE ACTION'
*****
* BC1PNM60 - CCLG
* AT THIS POINT, THE API BUFFER CONTAINS THE "OPLG" CMD FOLLOWED *
* BY THE "CLEAR" CMD. THE OLCAT AND OLLOC "SET" COMMANDS *
* ARE ALREADY PLACED IN THE BUFFER. *
*
* THIS ROUTINE WILL ADD THE FOLLOWING "SET" CMDS TO THE TEXT *
* PORTION OF THE RECORD.
* SET OLTEXT(1)='PACKAGE PROCESSING COMPLETED RC = PECBNDRC';
*           WHERE PECBNDRC = PACKAGE RETURN CODE VALUE.
* SET OLTEXT(2)='NETMAN ACCESSED FROM PACKAGE EXIT PECBBANM
*           PECBFNNM';
*           WHERE PECBBANM = EITHER BEFORE OR AFTER.
*           WHERE PECBFNNM = PACKAGE FUNCTION.
* SET OLTEXT(3)='PKG STATUS: PHDRSTAT USER: PECBUSER';
*           WHERE PHDRSTAT = PACKAGE STATUS.
*           WHERE PECBUSER = USER ID.
*****
CCLGRUTN DS    0H
      STM  R12,R14,WAL_CCLGSAVE
      LR   R12,R15
      USING CCLGRUTN,R12

```

```

*****
* COMPUTE TOTAL STRING LENGTH OF THE FIRST 3 TEXT LINES. *
*****
LA R2,L'CCLG_L1+8+L'CCLG_L2      LINE 1 LENGTH
LA R2,L'CCLG_L3+17+L'CCLG_L2(,R2) ADD LINE 2 LENGTH
LA R2,L'CCLG_L4+12(,R2)          ADD 1ST PART OF LINE 3
LA R2,L'CCLG_L5+8+L'CCLG_L2(,R2) ADD 2ND PART OF LINE 3
CH R2,WA1_BUFF_LEN               IF BUFFER TOO SMALL FOR
BH CCLGEXIT                      DATA, RETURN.
L  R4,WA1_DATALEN_ADDR           UPDTE CALLER'S AREA WITH
STH R2,0(,R4)                     TOTAL DATA LENGTH.
SPACE 1

* PLACE 1ST SET STRING INTO BUFFER.
L  R4,0(,R8)                     BUT FIRST ....
USING $PECBDS,R4                 CONVERT VARIABLE.
XR R1,R1
ICM R1,3,PECBNDRC               GET PACKAGE RETURN CODE
DROP R4
CVD R1,WA1_DBL                  NOW CONVERT IT FROM
OI WA1_DBL+7,X'0F'              BINARY TO DISPLAY
UNPK WA1_TARGET1,WA1_DBL
MVC WA1_TARGET2,CCLG_BLANKS    GET RID OF LEADING
LA R1,WA1_TARGET1              CHARACTER ZEROES.
LA R2,L'WA1_TARGET1
CCLG0110 DS 0H
CLI 0(R1),C'0'                  PUT RESULT OF RC
BNE CCLG0112                   IN TARGET2, PAD W/
LA R1,1(,R1)
BCT R2,CCLG0110
MVI WA1_TARGET2,C'0'
BZ CCLG0114

CCLG0112 DS 0H
BCTR R2,0
EX R2,CCLG_MVC1
B CCLG0114

CCLG_MVC1 DS 0H
MVC WA1_TARGET2(*--),0(R1)

```

```

CCLG0114 DS    0H
               L   R3,WA1_BUFF_ADDR      R3 -> BUFFER
               MVC 0(L'CCLG_L1,R3),CCLG_L1 MOVE STRING 1 INTO BUFFER
               LA   R3,L'CCLG_L1(,R3)    BUMP BUFFER BY STR LEN
               MVC 0(L'WA1_TARGET2,R3),WA1_TARGET2 MOVE IN VARIABLE
               LA   R3,L'WA1_TARGET2(,R3)  BUMP BUFFER BY VAR LEN
               MVC 0(L'CCLG_L2,R3),CCLG_L2 MOVE DEL STRING INTO BUF
               LA   R3,L'CCLG_L2(,R3)    BUMP BUFFER BY DEL LEN
               SPACE 1

*   PLACE 2ND SET STRING INTO BUFFER.
               MVC 0(L'CCLG_L3,R3),CCLG_L3 MOVE STRING 2 INTO BUFFER
               LA   R3,L'CCLG_L3(,R3)    BUMP BUFFER BY STR LEN
               L   R4,0(,R8)              R4-> $PECBDS EXIT C/B
               USING $PECBDS,R4
               MVC 0(L'PECBBANM,R3),PECBBANM MOVE EXIT TYPE VAR INTO BUF
               MVI L'PECBBANM(R3),C' '
               LA   R3,L'PECBBANM+1(,R3)  BUMP BUFFER BY VAR LEN
               MVC 0(L'PECBFNNM,R3),PECBFNNM MOVE IN EXIT NME VAR INTO BUF
               LA   R3,L'PECBFNNM(,R3)    BUMP BUFFER BY VAR LEN
               DROP R4
               MVC 0(L'CCLG_L2,R3),CCLG_L2 MOVE DEL STRING INTO BUF
               LA   R3,L'CCLG_L2(,R3)    BUMP BUFFER BY DEL LEN
               SPACE 1

*   PLACE 3RD SET STRING INTO BUFFER.
               MVC 0(L'CCLG_L4,R3),CCLG_L4 MOVE STRING 3 INTO BUFFER
               LA   R3,L'CCLG_L4(,R3)    BUMP BUFFER BY STR LEN
               L   R4,0(,R8)              R4-> $PECBDS EXIT C/B
               USING $PECBDS,R4
               L   R5,8(,R8)              R5-> $PHDRDS EXIT C/B
               USING $PHDRDS,R5
               MVC 0(L'PHDRSTAT,R3),PHDRSTAT MOVE STAT VAR INTO BUFFER
               CLC PECBFNNM,=CL8'DELETE' CHANGE STATUS TO N/A
               BNE CCLG0120                WHEN DELETE ACTION
               MVC 0(12,R3),=CL12'N/A'

```

```

CCLG0120 DS   0H
    LA R3,L'PHDRSTAT(,R3)      BUMP BUFFER BY VAR LEN
    MVC 0(L'CCLG_L5,R3),CCLG_L5 MOVE SUB-STR INTO BUFFER
    LA R3,L'CCLG_L5(,R3)      BUMP BUFFER BY STR LEN
    MVC 0(L'PECBUSER,R3),PECBUSER MOVE USER VAR INTO BUFFER
    LA R3,L'PECBUSER(,R3)      BUMP BUFER BY VAR LEN
    MVC 0(L'CCLG_L2,R3),CCLG_L2 MOVE DEL STRING INTO BUF
    LA R3,L'CCLG_L2(,R3)      BUMP BUFFER BY DEL LEN
    DROP R4,R5
    EJECT
*****
* SINCE THE 4TH STRING IS DEPENDENT ON THE PKG ACTION, BRANCH TO *
* PROPER ROUTINE.                                                 *
*****
L   R4,0(,R8)          R4-> $PECBDS EXIT C/B
USING $PECBDS,R4
LH   R1,PECBFNCD      GET FUNC NUMBER
SLL  R1,2              COMPUTE FUNC ENTRY SLOT
B    CCLGTABL(R1)      GO TO PROPER BRANCH ENTRY
SPACE 1
CCLGTABL DS  0H
    B   CCLGEXIT      SETUP EXIT OPTION
    B   CCLG0400      BACK-IN PACKAGE
    B   CCLG0350      BACK-OUT PACKAGE
    B   CCLG0250      CAST PACKAGE
    B   CCLG0450      COMMIT PACKAGE
    B   CCLG0200      CREATE PACKAGE
    B   CCLGEXIT      DELETE PACKAGE
    B   CCLGEXIT      DISPLAY PACKAGE
    B   CCLG0300      EXECUTE PACKAGE
    B   CCLGEXIT      EXPORT PACKAGE
    B   CCLGEXIT      GENERATE PACKAGE ID
    B   CCLGEXIT      LIST GENERATION
    B   CCLG0200      MODIFY PACKAGE
    B   CCLG0500      RESET PACKAGE
    B   CCLGEXIT      REVIEW PACKAGE
    B   CCLG0150      SHIP PACKAGE
    B   CCLGEXIT      ARCHIVE PACKAGE
    B   CCLGEXIT      COLLECT PRODUCT (DISTRIBLINK)
SPACE 1

```

```

CCLG0150 DS    0H
    CLC  PECBSFCD,=H'12'
    BE   CCLG0550           SHIP XMIT
    B    CCLG0600           SHIP CONFIRM
    DROP R4
    EJECT

*****
*   FOR "CREATE" AND "MODIFY" PACKAGE ACTIONS. *
*   SET OLTEXT(4)='CREATE: PHDRCRD PHDRCRT';
*   WHERE PHDRCRD = PACKAGE CREATE DATE.      *
*   WHERE PHDRCRT = PACKAGE CREATE TIME.      *
*****
SPACE 1
*****
* COMPUTE TOTAL STRING LENGTH INCLUDING THE 1ST 3 LINE. *
*****
CCLG0200 DS    0H
    LA   R2,L'CCLG_L6+L'PHDRCRD+L'PHDRCRT+1+L'CCLG_L2
    L    R4,WA1_DATALEN_ADDR      GET ADR OF USER'S DATALEN
    AH  R2,0(,R4)                ADD USER LENGTH W/ INTENDED
    CH  R2,WA1_BUFF_LEN         IF BUFFER TOO SMALL FOR
    BH  CCLGEXIT                DATA, RETURN.
    STH R2,0(,R4)                UPDATE TOTAL DATA LENGTH.
    SPACE 1
* PLACE 4TH SET STRING INTO BUFFER.
    MVC  0(L'CCLG_L6,R3),CCLG_L6      MOVE STRING 4 INTO BUFFER
    LA   R3,L'CCLG_L6(,R3)          BUMP BUFFER BY STR LEN
    L    R4,8(,R8)                 R4-> $PHDRDS EXIT C/B
    USING $PHDRDS,R4
    MVC  0(L'PHDRCRD,R3),PHDRCRD    MOVE DATE VAR INTO BUF
    MVI L'PHDRCRD(R3),C' '
    LA   R3,L'PHDRCRD+1(,R3)       ADD A SPACE AFTER VAR
    MVC  0(L'PHDRCRT,R3),PHDRCRT    BUMP BUFFER BY VAR LEN
    LA   R3,L'PHDRCRT(,R3)        MOVE TIME VAR INTO BUF
    DROP R4                         BUMP BUFFER BY VAR LEN
    MVC  0(L'CCLG_L2,R3),CCLG_L2    MOVE DEL STRING INTO BUF
    LA   R3,L'CCLG_L2(,R3)          BUMP BUFFER BY DEL LEN
    B    CCLGEXIT
    EJECT

```

```

*****
*   FOR "CAST" PACKAGE ACTION. *
*       SET OLTEXT(4)='CAST: PHDRCD PHDRCT';
*           WHERE PHDRCD    = PACKAGE CAST DATE. *
*           WHERE PHDRCT    = PACKAGE CAST TIME. *
*****
SPACE 1
*****
* COMPUTE TOTAL STRING LENGTH INCLUDING THE FIRST 3 TEXT LINE. *
*****
CCLG0250 DS 0H
    LA R2,L'CCLG_L7+L'PHDRCD+L'PHDRCT+1+L'CCLG_L2
    L  R4,WA1_DATALEN_ADDR      GET ADR OF USER'S DATALEN
    AH R2,0(,R4)                ADD USER LENGTH W/ INTENDED
    CH R2,WA1_BUFF_LEN         IF BUFFER TOO SMALL FOR
    BH CCLGEXIT                DATA, RETURN.
    STH R2,0(,R4)              UPDATE TOTAL DATA LENGTH.
    SPACE 1
* PLACE 4TH SET STRING INTO BUFFER.
    MVC 0(L'CCLG_L7,R3),CCLG_L7      MOVE STRING 4 INTO BUFFER
    LA R3,L'CCLG_L7(,R3)            BUMP BUFFER BY STR LEN
    L  R4,8(,R8)                  R4-> $PHDRDS EXIT C/B
    USING $PHDRDS,R4
    MVC 0(L'PHDRCD,R3),PHDRCD      MOVE DATE VAR INTO BUF
    MVI L'PHDRCD(R3),C' '
    LA R3,L'PHDRCD+1(,R3)        BUMP BUFFER BY VAR LEN
    MVC 0(L'PHDRCT,R3),PHDRCT      MOVE TIME VAR INTO BUF
    LA R3,L'PHDRCT(,R3)          BUMP BUFFER BY VAR LEN
    DROP R4
    MVC 0(L'CCLG_L2,R3),CCLG_L2      MOVE DEL STRING INTO BUF
    LA R3,L'CCLG_L2(,R3)          BUMP BUFFER BY DEL LEN
    B  CCLGEXIT
    EJECT
*****
*   FOR "EXECUTE" PACKAGE ACTION. *
*       SET OLTEXT(4)='EXECUTION: PHDRXD PHDRXT';
*           WHERE PHDRXD    = PACKAGE START EXECUTION DATE. *
*           WHERE PHDRXT    = PACKAGE START EXECUTION TIME. *
*       SET OLTEXT(5)='END EXECUTION: PHDREEXD PHDREEXT';
*           WHERE PHDREEXD   = PACKAGE END EXECUTION DATE. *
*           WHERE PHDREEXT   = PACKAGE END EXECUTION TIME. *
*****
SPACE 1

```

```
*****
* COMPUTE TOTAL STRING LENGTH INCLUDING THE FIRST 3 TEXT LINE. *
*****
```

CCLG0300 DS 0H
 LA R2,L'CCLG_L8+L'PHDRXD+L'PHDRXT+1+L'CCLG_L2
 LA R2,L'CCLG_L9+L'PHDREEXT+1+L'CCLG_L2(,R2)
 L R4,WA1_DATALEN_ADDR GET ADR OF USER'S DATALEN
 AH R2,0(,R4) ADD USER LENGTH W/ INTENDED
 CH R2,WA1_BUFF_LEN IF BUFFER TOO SMALL FOR
 BH CCLGEXIT DATA, RETURN.
 STH R2,0(,R4) UPDATE TOTAL DATA LENGTH.
 SPACE 1

* PLACE 4TH SET STRING INTO BUFFER.
 MVC 0(L'CCLG_L8,R3),CCLG_L8 MOVE STRING 4 INTO BUFFER
 LA R3,L'CCLG_L8(,R3) BUMP BUFFER BY STR LEN
 L R4,8(,R8) R4-> \$PHDRDS EXIT C/B
 USING \$PHDRDS,R4
 MVC 0(L'PHDRXD,R3),PHDRXD MOVE DATE VAR INTO BUF
 MVI L'PHDRXD(R3),C' '
 LA R3,L'PHDRXD+1(,R3) BUMP BUFFER BY VAR LEN
 MVC 0(L'PHDRXT,R3),PHDRXT MOVE TIME VAR INTO BUF
 LA R3,L'PHDRXT(,R3) BUMP BUFFER BY VAR LEN
 DROP R4
 MVC 0(L'CCLG_L2,R3),CCLG_L2 MOVE DEL STRING INTO BUF
 LA R3,L'CCLG_L2(,R3) BUMP BUFFER BY DEL LEN
 SPACE 1

* PLACE 5TH SET STRING INTO BUFFER.
 MVC 0(L'CCLG_L9,R3),CCLG_L9 MOVE STRING 5 INTO BUFFER
 LA R3,L'CCLG_L9(,R3) BUMP BUFFER BY STR LEN
 L R4,8(,R8) R4-> \$PHDRDS EXIT C/B
 USING \$PHDRDS,R4
 MVC 0(L'PHDREEXT,R3),PHDREEXT MOVE DATE VAR INTO BUF
 MVI L'PHDREEXT(R3),C' '
 LA R3,L'PHDREEXT+1(,R3) BUMP BUFFER BY VAR LEN
 MVC 0(L'PHDREEXT,R3),PHDREEXT MOVE TIME VAR INTO BUF
 LA R3,L'PHDREEXT(,R3) BUMP BUFFER BY VAR LEN
 DROP R4
 MVC 0(L'CCLG_L2,R3),CCLG_L2 MOVE DEL STRING INTO BUF
 LA R3,L'CCLG_L2(,R3) BUMP BUFFER BY DEL LEN
 B CCLGEXIT
 EJECT

```

*****
*   FOR "BACKOUT" PACKAGE ACTION. *
*       SET OLTEXT(4)='BACKOUT: PHDRBOD PHDRBOT STATUS: PHDRBOST'; *
*           WHERE PHDRBOD = PACKAGE BACKOUT DATE. *
*           WHERE PHDRBOT = PACKAGE BACKOUT TIME. *
*           WHERE PHDRBOST = PACKAGE BACKOUT STATUS. *
*****
SPACE 1
*****
* COMPUTE TOTAL STRING LENGTH INCLUDING THE FIRST 3 TEXT LINE. *
*****
CCLG0350 DS    0H
LA R2,L'CCLG_L10+L'PHDRBOD+L'PHDRBOT+1
LA R2,L'CCLG_L11+L'PHDRBOST+L'CCLG_L2(,R2)
L  R4,WA1_DATALEN_ADDR      GET ADR OF USER'S DATALEN
AH R2,0(,R4)                ADD USER LENGTH W/ INTENDED
CH R2,WA1_BUFF_LEN          IF BUFFER TOO SMALL FOR
BH CCLGEXIT                DATA, RETURN.
STH R2,0(,R4)               UPDATE TOTAL DATA LENGTH.
SPACE 1
* PLACE 4TH SET STRING INTO BUFFER.
MVC 0(L'CCLG_L10,R3),CCLG_L10 MOVE STRING 4 INTO BUFFER
LA R3,L'CCLG_L10(,R3)        BUMP BUFFER BY STR LEN
L  R4,8(,R8)                 R4-> $PHDRDS EXIT C/B
USING $PHDRDS,R4
MVC 0(L'PHDRBOD,R3),PHDRBOD MOVE DATE VAR INTO BUF
MVI L'PHDRBOD(R3),C' '
LA R3,L'PHDRBOD+1(,R3)      ADD A SPACE AFTER VAR
MVC 0(L'PHDRBOT,R3),PHDRBOT MOVE TIME VAR INTO BUF
LA R3,L'PHDRBOT(,R3)        BUMP BUFFER BY VAR LEN
MVC 0(L'CCLG_L11,R3),CCLG_L11 MOVE SUBSTR INTO BUFFER
LA R3,L'CCLG_L11(,R3)       BUMP BUFFER BY STR LEN
MVC 0(L'PHDRBOST,R3),PHDRBOST MOVE STATUS VAR INTO BUF
LA R3,L'PHDRBOST(,R3)       BUMP BUFFER BY VAR LEN
DROP R4
MVC 0(L'CCLG_L2,R3),CCLG_L2 MOVE DEL STRING INTO BUF
LA R3,L'CCLG_L2(,R3)        BUMP BUFFER BY DEL LEN
B  CCLGEXIT
EJECT
*****
* FOR "BACKIN" PACKAGE ACTION. *
*       SET OLTEXT(4)='BACKIN: PHDRBID PHDRBIT'; *
*           WHERE PHDRBID = PACKAGE BACKIN DATE. *
*           WHERE PHDRBIT = PACKAGE BACKIN TIME. *
*****
SPACE 1

```

```

*****
* COMPUTE TOTAL STRING LENGTH INCLUDING THE FIRST 3 TEXT LINE. *
*****
CCLG0400 DS    0H
    LA  R2,L'CCLG_L12+L'PHDRBID+L'PHDRBIT+1+L'CCLG_L2
    L   R4,WA1_DATALEN_ADDR      GET ADR OF USER'S DATALEN
    AH R2,0(,R4)                 ADD USER LENGTH W/ INTENDED
    CH R2,WA1_BUFF_LEN          IF BUFFER TOO SMALL FOR
    BH CCLGEXIT                DATA, RETURN.
    STH R2,0(,R4)               UPDATE TOTAL DATA LENGTH.
    SPACE 1

* PLACE 4TH SET STRING INTO BUFFER.
    MVC 0(L'CCLG_L12,R3),CCLG_L12 MOVE STRING 4 INTO BUFFER
    LA  R3,L'CCLG_L12(,R3)       BUMP BUFFER BY STR LEN
    L   R4,8(,R8)                R4-> $PHDRDS EXIT C/B
    USING $PHDRDS,R4
    MVC 0(L'PHDRBID,R3),PHDRBID MOVE DATE VAR INTO BUF
    MVI L'PHDRBID(R3),C' '
    LA  R3,L'PHDRBID+1(,R3)     BUMP BUFFER BY VAR LEN
    MVC 0(L'PHDRBIT,R3),PHDRBIT MOVE TIME VAR INTO BUF
    LA  R3,L'PHDRBIT(,R3)       BUMP BUFFER BY VAR LEN
    DROP R4
    MVC 0(L'CCLG_L2,R3),CCLG_L2 MOVE DEL STRING INTO BUF
    LA  R3,L'CCLG_L2(,R3)       BUMP BUFFER BY DEL LEN
    B   CCLGEXIT
    EJECT

*****
* FOR "COMMIT" PACKAGE ACTION. *
*   SET OLTEXT(4)='COMMIT: PHDRCMD PHDRCMT'; *
*   WHERE PHDRCMD = PACKAGE COMMIT DATE. *
*   WHERE PHDRCMT = PACKAGE COMMIT TIME. *
*****
    SPACE 1
*****
* COMPUTE TOTAL STRING LENGTH INCLUDING THE FIRST 3 TEXT LINE. *

```

```

*****
CCLG0450 DS   0H
    LA R2,L'CCLG_L13+L'PHDRCMD+L'PHDRCMT+1+L'CCLG_L2
    L  R4,WA1_DATALEN_ADDR      GET ADR OF USER'S DATALEN
    AH R2,0(,R4)                ADD USER LENGTH W/ INTENDED
    CH R2,WA1_BUFF_LEN         IF BUFFER TOO SMALL FOR
    BH CCLGEXIT                DATA, RETURN.
    STH R2,0(,R4)              UPDATE TOTAL DATA LENGTH.
    SPACE 1

*  PLACE 4TH SET STRING INTO BUFFER.
    MVC 0(L'CCLG_L13,R3),CCLG_L13 MOVE STRING 4 INTO BUFFER
    LA R3,L'CCLG_L13(,R3)        BUMP BUFFER BY STR LEN
    L  R4,8(,R8)                R4-> $PHDRDS EXIT C/B
    USING $PHDRDS,R4
    MVC 0(L'PHDRCMD,R3),PHDRCMD MOVE DATE VAR INTO BUF
    MVI L'PHDRCMD(R3),C' '
    LA R3,L'PHDRCMD+1(,R3)      ADD A SPACE AFTER VAR
    MVC 0(L'PHDRCMT,R3),PHDRCMT MOVE TIME VAR INTO BUF
    LA R3,L'PHDRCMT(,R3)        BUMP BUFFER BY VAR LEN
    DROP R4
    MVC 0(L'CCLG_L2,R3),CCLG_L2 MOVE DEL STRING INTO BUF
    LA R3,L'CCLG_L2(,R3)        BUMP BUFFER BY DEL LEN
    B   CCLGEXIT
    EJECT

*****
*  FOR "RESET" PACKAGE ACTION. *
*      SET OLTEXT(4)='RESET: DATE TIME'; *
*          WHERE DATE      = CURRENT DATE. *
*          WHERE TIME      = CURRENT TIME. *
*****
    SPACE 1

```

```
*****
* COMPUTE TOTAL STRING LENGTH INCLUDING THE FIRST 3 TEXT LINE. *
*****
CCLG0500 DS    0H
    LA  R2,L'CCLG_L14+L'WA1_CURR_DATE+L'WA1_CURR_TIME+1
    LA  R2,L'CCLG_L2(,R2)
    L   R4,WA1_DATALEN_ADDR      GET ADR OF USER'S DATALEN
    AH  R2,0(,R4)                ADD USER LENGTH W/ INTENDED
    CH  R2,WA1_BUFF_LEN         IF BUFFER TOO SMALL FOR
    BH  CCLGEXIT                DATA, RETURN.
    STH R2,0(,R4)                UPDATE TOTAL DATA LENGTH.
    SPACE 1

* PLACE 4TH SET STRING INTO BUFFER.
    MVC  0(L'CCLG_L14,R3),CCLG_L14  MOVE STRING 4 INTO BUFFER
    LA   R3,L'CCLG_L14(,R3)        BUMP BUFFER BY STR LEN
    MVC  0(L'WA1_CURR_DATE,R3),WA1_CURR_DATE  PLACE CURR DATE
    MVI L'WA1_CURR_DATE(R3),C' '   ADD A SPACE AFTER VAR
    LA   R3,L'WA1_CURR_DATE+1(,R3) BUMP BUFFER BY VAR LEN
    MVC  0(L'WA1_CURR_TIME,R3),WA1_CURR_TIME  PLACE CURR TIME
    LA   R3,L'WA1_CURR_TIME(,R3)  BUMP BUFFER BY VAR LEN
    MVC  0(L'CCLG_L2,R3),CCLG_L2  MOVE DEL STRING INTO BUF
    LA   R3,L'CCLG_L2(,R3)        BUMP BUFFER BY DEL LEN
    B    CCLGEXIT
    EJECT
*****
* FOR "SHIP XMIT" PACKAGE ACTION. *
* SET OLTEXT(4)='SHIP: DESTINATION PREQDEST TYPE PREQSTYP      *
*                 COMPLEMENTS PREQSCMP'; *
* WHERE PREQDEST = PACKAGE SHIP DESTINATION. *
* WHERE PREQSTYP = PACKAGE SHIPMENT TYPE. *
* WHERE PREQSCMP = SHIP COMPLEMENTS (EITHER Y/N). *
*****
    SPACE 1
```

```
*****
* COMPUTE TOTAL STRING LENGTH INCLUDING THE FIRST 3 TEXT LINE. *
*****
CCLG0550 DS   0H
    LA R2,L'CCLG_L15+L'PREQDEST
    LA R2,L'CCLG_L16+L'PREQSTYP(,R2)
    LA R2,L'CCLG_L17+L'PREQSCMP(,R2)
    LA R2,L'CCLG_L2(,R2)
    L  R4,WA1_DATALEN_ADDR      GET ADR OF USER'S DATALEN
    AH R2,0(,R4)                ADD USER LENGTH W/ INTENDED
    CH R2,WA1_BUFF_LEN         IF BUFFER TOO SMALL FOR
    BH CCLGEXIT                DATA, RETURN.
    STH R2,0(,R4)              UPDATE TOTAL DATA LENGTH.
    SPACE 1

* PLACE 4TH SET STRING INTO BUFFER.
    MVC 0(L'CCLG_L15,R3),CCLG_L15 MOVE STRING 4 INTO BUFFER
    LA R3,L'CCLG_L15(,R3)        BUMP BUFFER BY STR LEN
    L  R4,4(,R8)                R4-> $PREQPDS
    USING $PREQPDS,R4
    MVC 0(L'PREQDEST,R3),PREQDEST MOVE DEST VAR IN BUFFER
    LA R3,L'PREQDEST(,R3)        BUMP BUFFER BY VAR LEN
    MVC 0(L'CCLG_L16,R3),CCLG_L16 MOVE SUBSTR INTO BUFFER
    LA R3,L'CCLG_L16(,R3)        BUMP BUFFER BY STR LEN
    MVC 0(L'PREQSTYP,R3),PREQSTYP MOVE STYPE VAR INTO BUF
    LA R3,L'PREQSTYP(,R3)        BUMP BUFFER BY VAR LEN
    MVC 0(L'CCLG_L17,R3),CCLG_L17 MOVE SUBSTR INTO BUFFER
    LA R3,L'CCLG_L17(,R3)        BUMP BUFFER BY STR LEN
    MVC 0(L'PREQSCMP,R3),PREQSCMP MOVE SCMP VAR INTO BUF
    LA R3,L'PREQSCMP(,R3)        BUMP BUFFER BY VAR LEN
    DROP R4
    MVC 0(L'CCLG_L2,R3),CCLG_L2  MOVE DEL STRING INTO BUF
    LA R3,L'CCLG_L2(,R3)        BUMP BUFFER BY DEL LEN
    B  CCLGEXIT
    EJECT
```

```

*****
* FOR "SHIP CONFIRM" PACKAGE ACTION. *
* SET OLTEXT(4)='SHIP: DESTINATION PREQDEST TYPE PREQSCNF *
*           RESULTS PREQSRES PREQSRCV'; *
* WHERE PREQDEST = PACKAGE SHIP DESTINATION. *
* WHERE PREQSCNF = PACKAGE SHIP CONFIRMATION TYPE. *
* WHERE PREQSRES = SHIP CONFIRMATION RESULTS. *
* WHERE PREQSRCV = SHIP CONFIRM RETURN CODE. *
*****
SPACE 1
*****
* COMPUTE TOTAL STRING LENGTH INCLUDING THE FIRST 3 TEXT LINE. *
*****
CCLG0600 DS   0H
    LA  R2,L'CCLG_L15+L'PREQDEST
    LA  R2,L'CCLG_L16+L'PREQSCNF(,R2)
    LA  R2,L'CCLG_L18+L'PREQSRES+L'PREQSRCV+1(,R2)
    LA  R2,L'CCLG_L2(,R2)
    L   R4,WA1_DATALEN_ADDR      GET ADR OF USER'S DATALEN
    AH  R2,0(,R4)                ADD USER LENGTH W/ INTENDED
    CH  R2,WA1_BUFF_LEN         IF BUFFER TOO SMALL FOR
    BH  CCLGEXIT                DATA, RETURN.
    STH R2,0(,R4)                UPDATE TOTAL DATA LENGTH.
    SPACE 1

```

```

* PLACE 4TH SET STRING INTO BUFFER.
    MVC 0(L'CCLG_L15,R3),CCLG_L15  MOVE STRING 4 INTO BUFFER
    LA  R3,L'CCLG_L15(,R3)      BUMP BUFFER BY STR LEN
    L   R4,4(,R8)                R4-> $PREQPDS
    USING $PREQPDS,R4
    MVC 0(L'PREQDEST,R3),PREQDEST  MOVE DEST VAR IN BUFFER
    LA  R3,L'PREQDEST(,R3)      BUMP BUFFER BY VAR LEN
    MVC 0(L'CCLG_L16,R3),CCLG_L16  MOVE SUBSTR INTO BUFFER
    LA  R3,L'CCLG_L16(,R3)      BUMP BUFFER BY STR LEN
    MVC 0(L'PREQSCNF,R3),PREQSCNF  MOVE CTYPE VAR INTO BUF
    LA  R3,L'PREQSCNF(,R3)      BUMP BUFFER BY VAR LEN
    MVC 0(L'CCLG_L18,R3),CCLG_L18  MOVE SUBSTR INTO BUFFER
    LA  R3,L'CCLG_L18(,R3)      BUMP BUFFER BY STR LEN
    MVC 0(L'PREQSRES,R3),PREQSRES  MOVE RESULT VAR INTO BUF
    MVI L'PREQSRES(R3),C' '
    LA  R3,L'PREQSRES+1(,R3)    BUMP BUFF BY VAR & SP LEN
    MVC 0(L'PREQSRCV,R3),PREQSRCV  MOVE RC VAR INTO BUF
    LA  R3,L'PREQSRCV(,R3)      BUMP BUFF BY VAR LEN
    DROP R4
    MVC 0(L'CCLG_L2,R3),CCLG_L2  MOVE DEL STRING INTO BUF
    LA  R3,L'CCLG_L2(,R3)      BUMP BUFFER BY DEL LEN
    B   CCLGEXIT
    SPACE 2
    CCLGEXIT DS 0H
    LM  R12,R14,WA1_CCLGSAVE
    XR  R15,R15
    BR  R14
    SPACE 1
    CCLG_L1 DC C'SET OLTEXT(1)='PACKAGE PROCESSING COMPLETED RC = '
    CCLG_L2 DC C''';;
    CCLG_L3 DC C'SET OLTEXT(2)='NETMAN ACCESSED FROM PACKAGE EXIT '
    CCLG_L4 DC C'SET OLTEXT(3)='PKG STATUS: '
    CCLG_L5 DC C' USER: '
    CCLG_L6 DC C'SET OLTEXT(4)='CREATE: '
    CCLG_L7 DC C'SET OLTEXT(4)='CAST: '
    CCLG_L8 DC C'SET OLTEXT(4)='EXECUTION: '
    CCLG_L9 DC C'SET OLTEXT(5)='END EXECUTION: '
    CCLG_L10 DC C'SET OLTEXT(4)='BACKOUT: '
    CCLG_L11 DC C' STATUS: '
    CCLG_L12 DC C'SET OLTEXT(4)='BACKIN: '
    CCLG_L13 DC C'SET OLTEXT(4)='COMMIT: '
    CCLG_L14 DC C'SET OLTEXT(4)='RESET: '
    CCLG_L15 DC C'SET OLTEXT(4)='SHIP: DESTINATION '
    CCLG_L16 DC C' TYPE '
    CCLG_L17 DC C' COMPLEMENTS '
    CCLG_L18 DC C' RESULTS '
    SPACE 1
    CCLG_BLANKS DC CL80'
    SPACE 1
    LTORG
    DROP R12
    EJECT
    END

```
