

AllFusion™ Endeavor® Change Manager

API Guide
4.0



Computer Associates®

SP1
ENAPI400

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.

Second 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. Welcome to the Application Program Interface	1-1
1.1 Overview	1-2
1.1.1 Assumed Knowledge	1-3
1.2 Endeavor API Architecture	1-4
1.3 API Structures	1-5
1.3.1 The Control Structure	1-5
1.3.2 The Request Structure	1-6
1.3.3 The Request Extension Structure	1-6
1.3.4 The Response Structure	1-6
1.4 API Function Calls	1-7
1.5 Initializing API Structures	1-12
1.5.1 Assembler Programs	1-12
1.5.2 COBOL Programs	1-12
1.6 Starting Up and Shutting Down the API Server	1-14
1.7 Calling the API from an Assembler Program	1-15
1.8 Calling the API from a COBOL Program	1-16
1.9 Checking API Return and Reason Codes	1-17
1.10 Writing Messages to the Message File	1-18
1.11 Writing Responses to a Response File	1-19
1.12 Sample Applications	1-20
1.13 Documentation Overview	1-21
1.14 Name Masking	1-22
1.14.1 Usage	1-22
Chapter 2. API Function Calls	2-1
2.1 Overview	2-2
2.2 Control Structure	2-3
2.2.1 AACTL Control Structure Fields	2-3
2.3 API Function Calls	2-8
2.3.1 Understanding Logical and Physical Mapping Requests	2-8
2.4 Request Extension	2-10
2.4.1.1 AAREB_RQ Request Extension Structure Fields	2-10
2.5 Add Element Action	2-11
2.5.1 AEADD_RQ Request Structure Fields	2-11
2.6 Delete Element Action	2-13
2.6.1 AEDEL_RQ Request Structure Fields	2-13
2.7 Extract Element and Component Data	2-15
2.7.1 Element and Component Extraction Types	2-15
2.7.2 AEELM_RQ Request Structure Fields	2-16

2.7.3	AEELM_RS Response Structure Fields	2-17
2.7.4	Element Extract and Component Data Record Layouts	2-17
2.7.4.1	Element Extract, No Format Record Layout	2-18
2.7.4.2	Element Extract, Browse Record Layout	2-19
2.7.4.3	Element Extract, Change Record Layout	2-19
2.7.4.4	Element Extract, History Record Layout	2-20
2.7.4.5	Component Extract, Browse Record Layout	2-21
2.7.4.6	Component Extract, Change Record Layout	2-22
2.7.4.7	Component Extract, History Record Layout	2-23
2.8	Generate Element Action	2-24
2.8.1	AEGEN_RQ Request Structure Fields	2-24
2.9	List Element	2-26
2.9.1	ALELM_RQ Request Structure Fields	2-26
2.9.1.1	Information About Action Options	2-27
2.9.1.2	Information About Selection Criteria	2-29
2.9.2	List Element Response Structures	2-31
2.9.3	ALELM_RS Response Structure Fields	2-32
2.9.3.1	Information About the Last Action	2-34
2.9.3.2	Information About the Element Base	2-34
2.9.3.3	Information About the Element Delta (Last Level)	2-34
2.9.3.4	Information About the Component List Base	2-35
2.9.3.5	Information About the Component List Delta	2-35
2.9.3.6	Information About the Last Element Move	2-36
2.9.3.7	Information About the Last Add or Update Data Set	2-36
2.9.3.8	Information About the Element Processor Execution	2-36
2.9.3.9	Information About the Last Element Retrieve	2-37
2.9.3.10	Information About the Package Last Executed Against the Element Source	2-38
2.9.3.11	Information About the Package Last Executed Against the Element Outputs	2-38
2.9.3.12	Information About the Last "FROM" Endeavor Location	2-38
2.9.3.13	Other Fields	2-39
2.9.3.14	Information About the Package for which the Element is Locked	2-39
2.9.3.15	ALELB_RS Response Structure Fields	2-39
2.9.3.16	ALELX_RS Response Structure Fields	2-41
2.10	Move Element Action	2-42
2.10.1	AEMOV_RQ Request Structure Fields	2-42
2.11	Print Element Action	2-44
2.11.1	AEPRE_RQ Request Structure Fields	2-44
2.12	Print Member Element Action	2-46
2.12.1	AEPRM_RQ Request Structure Fields	2-46
2.13	Retrieve Element Action	2-47
2.13.1	AERET_RQ Request Structure Fields	2-47
2.14	Signin Element Action	2-49
2.14.1	AESIG_RQ Request Structure Fields	2-49
2.15	Transfer Element Action	2-51
2.15.1	AETRA_RQ Request Structure Fields	2-51
2.16	Update Element Action	2-54
2.16.1	AEUPD_RQ Request Structure Fields	2-54
2.17	Enterprise Package Function	2-56
2.17.1	Enterprise Package Fields	2-56

2.17.2	Enterprise Package Restrictions	2-58
2.18	Approve Package	2-60
2.18.1	APAPP_RQ Request Structure Fields	2-60
2.19	Backin Package	2-61
2.19.1	APBKI_RQ Request Structure Fields	2-61
2.20	Backout Package	2-62
2.20.1	APBKO_RQ Request Structure Fields	2-62
2.21	Cast Package	2-63
2.21.1	APCAS_RQ Request Structure Fields	2-63
2.22	Commit Package	2-66
2.22.1	APCOM_RQ Request Structure Fields	2-66
2.23	Define Package	2-67
2.23.1	APDEF_RQ Request Structure Fields	2-67
2.23.1.1	Note Fields	2-69
2.24	Delete Package	2-70
2.24.1	APDEL_RQ Request Structure Fields	2-70
2.25	Deny Package	2-71
2.25.1	APDEN_RQ Request Structure Fields	2-71
2.26	Execute Package	2-72
2.26.1	APEXE_RQ Request Structure Fields	2-72
2.27	List Package Action Summary	2-73
2.27.1	ALSUM_RQ Request Structure Fields	2-73
2.27.2	ALSUM_RS Response Structure Fields	2-73
2.27.2.1	Source Location Information	2-74
2.27.2.2	Data Available When Location is C or A	2-74
2.27.2.3	Data Available When Location is D, F or P	2-75
2.27.2.4	Target Location Information	2-75
2.27.2.5	Data Available When Location is C or A	2-76
2.27.2.6	Data Available When Location is D, F or P	2-77
2.27.2.7	Data Available When Location is: C, A, D, F or P	2-77
2.28	List Package Approvers	2-78
2.28.1	ALAPP_RQ Request Structure Fields	2-78
2.28.2	ALAPP_RS Response Structure Fields	2-78
2.29	List Package Backout Information	2-80
2.29.1	ALBKO_RQ Request Structure Fields	2-80
2.29.2	ALBKO_RS Response Structure Fields	2-80
2.30	List Package Cast Report	2-82
2.30.1	ALCAS_RQ Request Structure Fields	2-82
2.30.2	ALCAS_RS Response Structure Fields	2-82
2.31	List Package Correlation	2-83
2.31.1	ALCOR_RQ Request Structure Fields	2-83
2.31.2	ALCOR_RS Response Structure Fields	2-83
2.32	List Package Header	2-85
2.32.1	ALPKG_RQ Request Structure Fields	2-85
2.32.2	List Package Response Structure Fields	2-89
2.32.3	ALPKG_RS Response Structure Fields	2-89
2.32.4	ALPKB_RS Response Structure Fields	2-92
2.33	List Package SCL	2-94
2.33.1	ALSCL_RQ Request Structure Fields	2-94
2.33.2	ALSCL_RS Response Structure Fields	2-94

2.34	Package Correlation	2-96
2.34.1	APCOR_RQ Request Structure Fields	2-96
2.35	Reset Package	2-97
2.35.1	APRES_RQ Request Structure Fields	2-97
2.36	Submit Package Request	2-98
2.36.1	APSUB_RQ Request Structure Fields	2-98
2.36.1.1	Jobcard Location Information	2-98
2.36.1.2	Submit TO Location Information	2-99
2.36.1.3	Action Options	2-99
2.36.1.4	CA7 Action Options	2-100
2.37	List Approver Group	2-101
2.37.1	ALAGR_RQ Request Structure Fields	2-101
2.37.2	ALAGR_RS Response Structure Fields	2-101
2.38	List Approver Group Junctions	2-103
2.38.1	ALAGJ_RQ Request Structure Fields	2-103
2.38.2	ALAGJ_RS Response Structure Fields	2-104
2.39	List Components/Where-used	2-105
2.39.1	ALCMP_RQ Request Structure Fields	2-105
2.39.1.1	Element Location Data – Request Type (E)	2-105
2.39.1.2	Member Location Data — Request Type (M)	2-106
2.39.1.3	Object Location Data — Request Type (O)	2-106
2.39.1.4	Comment Location Data — Request Type (C)	2-106
2.39.1.5	Output Filters — Request Type (C, E, M, O)	2-107
2.39.1.6	Build Generate Action SCL	2-107
2.39.2	ALCMP_RS Response Structure Fields	2-108
2.39.2.1	RECTYP 1 or 3 Format	2-109
2.39.2.2	RECTYP 2 or 4 Format	2-110
2.39.2.3	RECTYP 5 or 6 Format	2-110
2.40	List Data Set	2-111
2.40.1	ALDSN_RQ Request Structure Fields	2-111
2.40.2	ALDSN_RS Response Structure Fields	2-112
2.41	List Directory	2-113
2.41.1	ALDIR_RQ Request Structure Fields	2-113
2.41.2	ALDIR_RS Response Structure Fields	2-114
2.42	List Environment	2-116
2.42.1	ALENV_RQ Request Structure Fields	2-116
2.42.2	ALENV_RS Response Structure Fields	2-117
2.43	List Processor Group	2-119
2.43.1	ALPGR_RQ Request Structure Fields	2-119
2.43.2	ALPGR_RS Response Structure Fields	2-120
2.44	List Site	2-122
2.44.1	ALSIT_RQ Request Structure Fields	2-122
2.44.2	ALSIT_RS Response Structure Fields	2-122
2.45	List Stage	2-126
2.45.1	ALSTG_RQ Request Structure Fields	2-126
2.45.2	ALSTG_RS Response Structure Fields	2-127
2.46	List Subsystem	2-128
2.46.1	ALSBS_RQ Request Structure Fields	2-128
2.46.2	ALSBS_RS Response Structure Fields	2-129
2.47	List System	2-131
2.47.1	ALSYS_RQ Request Structure Fields	2-131

2.47.2	ALSYS_RS Response Structure Fields	2-132
2.48	List Type	2-134
2.48.1	ALTYR_RQ Request Structure Fields	2-134
2.48.2	ALTYR_RS Response Structure Fields	2-135
Chapter 3.	API Execution Reports and Trace Facilities	3-1
3.1	Overview	3-2
3.1.1	Execution Reports	3-2
3.1.2	Trace Facilities	3-2
3.2	API Execution Reports	3-3
3.2.1	Define Package Action Function Call Sample Report	3-3
3.2.2	Element Action Function Call Sample Report	3-4
3.2.3	Inventory List Function Call Sample Report	3-5
3.2.4	List Package Action Function Call Sample Reports	3-5
3.2.5	Update Package Action Function Call Sample Reports	3-7
3.3	The API Diagnostic Trace — BC1PAPI	3-9
3.4	The API Internal Trace — EN\$TRAPI	3-10
Appendix A.	Sample API Programs	A-1
A.1	Overview	A-2
A.2	Executing an API Program	A-3
A.2.1	Description	A-3
A.3	Sample COBOL Program — CCIDRPT1	A-5
A.3.1	Description	A-5
A.3.2	CCIDRPT1 Output Report	A-6
A.3.3	JCL to Execute CCIDRPT1 — BC1JRAPI	A-7
A.4	Sample List Environment Function Call — ENHAAPGM	A-10
A.4.1	Description	A-10
A.4.2	JCL to Execute ENHAAPGM — BC1JAPGM	A-10
A.5	Sample Element Action Function Call — ENHAEPGM	A-12
A.5.1	Description	A-12
A.5.2	JCL to Execute ENHAEPGM — BC1JEPGM	A-12
A.6	Sample List Package Action Function Call — ENHAPLST	A-15
A.6.1	Description	A-15
A.6.2	JCL to Execute ENHAPLST — BC1JPLST	A-15
A.7	Sample Update Package Action Function Call — ENHAPUPD	A-17
A.7.1	Description	A-17
A.7.2	JCL to Execute ENHAPUPD — BC1JPUPD	A-17
A.8	Sample Inventory List Function Call — ENTBJAPI	A-19
A.8.1	Description	A-19
A.8.2	JCL to Execute ENTBJAPI — BC1JAAPI	A-19
Appendix B.	Index	B-1

Chapter 1. Welcome to the Application Program Interface

1.1 Overview

Endevor is a comprehensive, automated solution for managing the entire software development life cycle. From initial design through distribution, it guarantees consistency and control through process automation and life cycle administration.

This document introduces the Endevor Application Program Interface (API). The Endevor API lets you retrieve and update information programmatically from Assembler or an LE-compliant language. This document describes how to use the API and describes sample programs and contains sample JCL to help you implement the API.

Endevor functions supported by the API can be categorized into four groups:

- **Inventory Query and List Functions**

Allows you to request inventory lists and perform queries. This includes elements, environments, systems, subsystems, and all other inventory information stored in the MCF.

- **Element Extract**

Allows you to extract element and/or component source from the base and delta libraries. Also, you can extract summary, changes, and history information associated with element or component data.

- **Element Actions**

Allows you to perform one of the element actions, such as ADD, GENERATE, MOVE, or PRINT ELEMENT. These functions retrieve information from the MCF and base and delta libraries, and may update these files, depending on the function.

The SCL SET OPTIONS clause is not supported. All SCL syntax is supported unless otherwise noted. For more information, see the *SCL Reference Guide*.

Actions not supported include:

- ARCHIVE
- COPY
- LIST from archive
- LIST member from external library
- RESTORE
- TRANSFER element to archive data set
- TRANSFER from archive data set or unload tape

- **Package Actions**

Allows you to perform package list and update actions such as approve, cast, delete or execute.

Masking is not allowed for any of the package update actions. You need to specify the fully qualify package id. The following list shows packages and the clauses they do not support:

- Approve does not support 'notes' clause.
- Commit does not support 'older than' clause.
- Delete does not support 'older than' and 'package status' clauses.
- Deny does not support 'notes' clause.
- Execute does not support 'execution window' and 'package status' clauses.

1.1.1 Assumed Knowledge

To use the Endeavor API, we assume that you are familiar with Endeavor. In addition, knowledge of COBOL or Assembler programming concepts is expected. Finally, you must be familiar with both the layout and runtime usage of structures; the API uses structures to request and receive information from Endeavor.

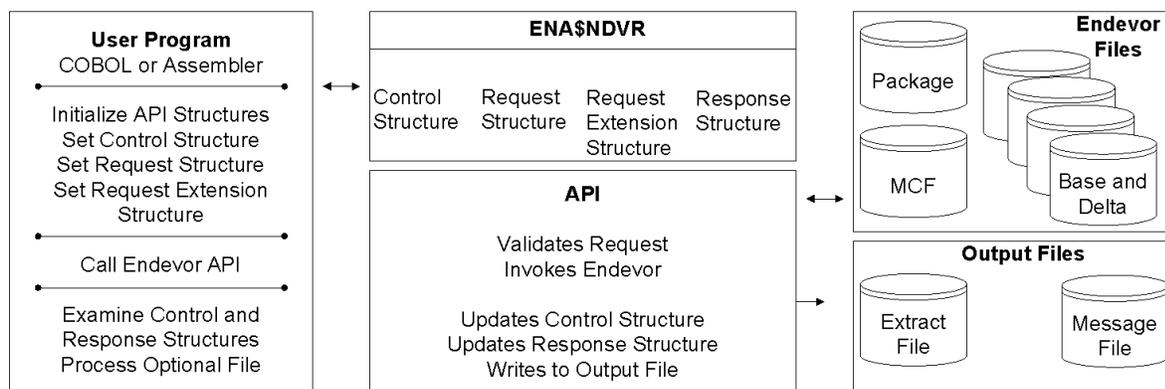
1.2 Endeavor API Architecture

To invoke the Endeavor API, an application program calls the Endeavor API interface program, ENA\$NDVR, passing it two, three or four parameters: the control structure, a request structure, and depending on the function call, a request extension structure and a response structure. List and extract function calls require all three structures. Element action function calls require the control and request structures.

Note: Your program that invokes the API must reside in an authorized library. No other restrictions apply to compiling or linking a user program.

On the first function call to the ENA\$NDVR interface program, the Endeavor API server is attached and initialized. The user's request is processed and the results are placed into the response structure if it exists. The control structure is also updated with return code and reason code information. The API server remains active and is available to process additional requests until a request is received to shutdown the API.

The following diagram shows how the Endeavor API processes an API function call:



You can also use the CONAPI utility to execute a program that issues API function calls from an Endeavor processor. For more information about this utility, see the *Extended Processors Guide*.

To execute a program that issues API function calls outside of a processor, see A.2, “Executing an API Program” on page A-3.

1.3 API Structures

There is one control structure used for all types of function calls and a unique request and response structure for each type of function call. Each structure consists of a header area, followed by a data area. It is the responsibility of the user program to initialize all the structures prior to calling the API and to populate some of the data area fields in the control and request structures.

The API Assembler macros and COBOL copybooks are provided in `ipfx.igual.SOURCE` for each API function call.

The structure naming conventions are as follows:

Name	Description
ExHAACTL	Control structure. Where <i>x</i> is N for Assembler macro and C for COBOL copybook.
ExHAEeee	Element actions. Where <i>x</i> is N for Assembler macro and C for COBOL copybook. Where <i>eee</i> is an element action such as Add, Generate or Delete.
ExHAPppp	Package actions. Where <i>x</i> is N for Assembler macro and C for COBOL copybook. Where <i>ppp</i> is a package action such as Define or Cast.
ExHALiep	List request. Where <i>x</i> is N for Assembler macro and C for COBOL copybook. Where <i>iep</i> is one of the inventory, element or package list actions.
ExHAAREB	Request Extension structure. Where <i>x</i> is N for Assembler macro and C for COBOL copybook.

1.3.1 The Control Structure

The control structure contains fields that:

- Start up and shutdown the API server.
- Define the message and response files where the information from each function call is placed. Each API function call opens and closes a response file. If two or more API function calls use the same response file, only the response to the last function call is maintained; data from the prior function call is overwritten.
- Contain the reason and return codes about the function call that you have performed.

For a description of the data areas included in the control structure, see 2.2, “Control Structure” on page 2-3.

1.3.2 The Request Structure

The request structure contains the fields required to perform an API function call. In this structure, you define location information and action options.

There is one unique request structure for each type of request. The calling program must populate the fields within this structure. The structure informs the API which action is being requested, provides the 'to and from' location specifications and indicates which action options are in effect. In cases where an action option has a default value and you intend to use the default, you do not need to code the option. Except for several noted cases, such as the SEARCH map option, if you do not code an option, the option is not enabled.

1.3.3 The Request Extension Structure

This request block defines the parameters necessary to process long name elements or HFS file structures. If you are not processing long name elements or HFS files, this block can be ignored. If specified, it must be the third parameter in the calling sequence to the API interface program, ENA\$NDVR, following the request block.

1.3.4 The Response Structure

The response structure contains header information followed by the data related to the request you made. This structure does not exist for element and package update type action calls. It is returned for list type action calls. One and only one response structure is returned to the calling program. The data returned in the response structure is unique for each type of request.

The response or list file DDN (AACTL_LIST_DDN) field defined in the Control Structure determines which response record is returned to the calling program in the response structure. If you specify a value in this field, the first response is returned to the calling program in the response structure.

Note: All dates return in DDMMYY (31JAN02) format and all times return in HH:MM (23:59) format.

1.4 API Function Calls

The Endeavor API provides Assembler macros and corresponding COBOL copybooks that correspond to the function calls. These are delivered in the `iprfx.igual.SOURCE` data set.

For each Assembler API macro, except the initialization macro, `API$INIT`, there is a corresponding COBOL copybook with a similar name. For example, the `ENHALENV` Assembler macro has a corresponding COBOL copybook of a similar name except that Assembler macros begin with `ENH` and COBOL copybooks begin with `ECH`. In addition, COBOL users have a copybook, `ENCCNST`, that contains Endeavor API constants.

The following table summarizes the types of API function calls that you can make and their corresponding structure names. Each extract and list function call has its own unique request and response structure. Response structures are not applicable for element action function calls. Where x is N for Assembler macros and C for COBOL copybooks.

API Function Call	Macro/Copybook Name	Request Structure Name	Response Structure Name
Add Element Defines the parameters necessary to add an element.	ExHAEADD	AEADD_RQ	N/A
Approve Package Defines the parameters necessary to approve a package.	ExHAPAPP	APAPP_RQ	N/A
Backin Package Defines the parameters necessary to backin a package.	ExHAPBKI	APBKI_RQ	N/A
Backout Package Defines the parameters necessary to backout a package.	ExHAPBKO	APBKO_RQ	N/A
Cast Package Defines the parameters necessary to cast a package.	ExHAPCAS	APCAS_RQ	N/A
Commit Package Defines the parameters necessary to commit a package.	ExHAPCOM	APCOM_RQ	N/A

1.4 API Function Calls

API Function Call	Macro/Copybook Name	Request Structure Name	Response Structure Name
Define Package Defines the parameters necessary to create or modify a package.	ExHAPDEF	APDEF_RQ	N/A
Delete element Defines the parameters necessary to delete an element.	ExHAEDEL	AEDEL_RQ	N/A
Delete Package Defines the parameters necessary to delete a package.	ExHAPDEL	APDEL_RQ	N/A
Deny Package Defines the parameters necessary to deny approval of a package.	ExHAPDEN	APDEN_RQ	N/A
Execute Package Defines the parameters necessary to execute a package.	ExHAPEXE	APEXE_RQ	N/A
Extract Element/Component Request Extracts the element source or element component information from an explicit Endeavor location and element level into your specified file.	ExHAEELM	AEELM_RQ	AEELM_RS
Generate Element Defines the parameters necessary to generate an element.	ExHAEGEN	AEGEN_RQ	N/A
List Approver Group Extracts approver group information for an environment.	ExHALAGR	ALAGR_RQ	ALAGR_RS
List Approver Group Junction Extracts approver junction information for an environment.	ExHALAGJ	ALAGJ_RQ	ALAGJ_RS
List Components Defines the parameters necessary to produce a component list for an element or a "where-used" list for a component.	ExHALCMP	ALCMP_RQ	ALCMP_RS

API Function Call	Macro/Copybook Name	Request Structure Name	Response Structure Name
List Dataset Extracts Endeavor data set information under an environment, stage, and system.	ExHALDSN	ALDSN_RQ	ALDSN_RS
List Directory Defines the parameters necessary to build a directory list of a file or to build a list of CSECTs for a load module.	ExHALDIR	ALDIR_RQ	ALDIR_RS
List Element Extracts element information for an environment and also mapped locations.	ExHALELM	ALELM_RQ	ALELM_RS
List Environment Extracts environment information and also mapped environments.	ExHALENV	ALENV_RQ	ALENV_RS
List Package Action Summary Defines the parameters necessary to list element actions associated with a package.	ExHALSUM	ALSUM_RQ	ALSUM_RS
List Package Approvers Defines the parameters necessary to produce a list of approvers for a package.	ExHALAPP	ALAPP_RQ	ALAPP_RS
List Package Backout Information Defines the parameters necessary to list the backout information associated with a package.	ExHALBKO	ALBKO_RQ	ALBKO_RS
List Package Cast Report Defines the parameters necessary to list a cast report associated with a package.	ExHALCAS	ALCAS_RQ	ALCAS_RS
List Package Correlation Defines the parameters necessary to list correlation records associated with a package.	ExHALCOR	ALCOR_RQ	ALCOR_RS

API Function Call	Macro/Copybook Name	Request Structure Name	Response Structure Name
List Package Header Defines the parameters necessary to provide a list of packages and the data associated with.	ExHALPKG	ALPKG_RQ	ALPKG_RS
List Package SCL Defines the parameters necessary to list the SCL associated with a package.	ExHALSCL	ALSCL_RQ	ALSCL_RS
List Processor Group Extracts processor information including the symbolic overrides under an environment, stage, system, and type.	ExHALPGR	ALPGR_RQ	ALPGR_RS
List Site Extracts the site's definition.	ExHALSIT	ALSIT_RQ	ALSIT_RS
List Stage Extracts stage information for an environment and also mapped stages.	ExHALSTG	ALSTG_RQ	ALSTG_RS
List Subsystem Extracts subsystem information for an environment and also mapped subsystems.	ExHALSBS	ALSBS_RQ	ALSBS_RS
List System Extracts system information for an environment and also mapped systems.	ExHALSYS	ALSYS_RQ	ALSYS_RS
List Type Extracts type information for an environment and also mapped types.	ExHALTYP	ALTYP_RQ	ALTYP_RS
Move Element Defines the parameters necessary to move an element.	ExHAEMOV	AEMOV_RQ	N/A
Package Correlation Defines the parameters necessary to create, delete or modify a correlation record associated with a package.	ExHAPCOR	APCOR_RQ	N/A

API Function Call	Macro/Copybook Name	Request Structure Name	Response Structure Name
Print Element Defines the parameters necessary to print an element.	ExHAEPRE	AEPRE_RQ	N/A
Print from External Library Defines the parameters necessary to print a member from an external library.	ExHAEPRM	AEPRM_RQ	N/A
Reset Package Defines the parameters necessary to reset a package.	ExHAPRES	APRES_RQ	N/A
Request Extension Defines the parameters necessary to process long name elements or HFS file structures.	ExHAAREB	AAREB_RQ	N/A
Retrieve Element Defines the parameters necessary to retrieve an element.	ExHAERET	AERET_RQ	N/A
Signin Element Defines the parameters necessary to sign in an element.	ExHAESIG	AESIG_RQ	N/A
Transfer Element Defines the parameters necessary to transfer an element.	ExHAETRA	AETRA_RQ	N/A
Update Element Defines the parameters necessary to update an element.	ExHAEUPD	AEUPD_RQ	N/A

COBOL Users: COBOL copybook field names are similar to the names used by the Assembler macros except that COBOL substitutes the underscore character (_) with a hyphen (-). For example, ALELM_RQ_PATH appears as ALELM-RQ-PATH.

See 2.9.1.1, “Information About Action Options” on page 2-27 for more information on this field name.

See Chapter 2, “API Function Calls” on page 2-1 for detailed information about each API function call and its use.

1.5 Initializing API Structures

Before you issue an API function call, you must first initialize the control, request, request extension and response structures passed with the API function call. It is the responsibility of the user program to initialize all the structures prior to calling the API and to populate some of the data area fields in the control and request structures.

The following sections explain how to initialize API structures in Assembler and COBOL programs.

1.5.1 Assembler Programs

Assembler users can use the `API$INIT` macro to initialize the control, request, and response structures. Each structure has a fixed formatted header and can be initialized by using `API$INIT`.

Note: The twelve-character header must not be initialized or modified by the user program. If it is, the API function call fails with an "invalid request structure" error condition. The initialization macro, `API$INIT` properly sets the header information.

To initialize the `ENHAACTL` control structure, use this statement:

```
API$INIT STG=AACTL,BLOCK=AACTL
```

To initialize a request or response structure, use this `API$INIT` syntax:

```
API$INIT STG=structure_name,BLOCK=structure_name
```

where `structure-name` is the name of the API macro request or response structure.

For example, these statements initialize the request and response structure for the list environment macro, `ENHLENV`:

```
API$INIT STG=ALENV_RQ,BLOCK=ALENV_RQ
API$INIT STG=ALENV_RS,BLOCK=ALENV_RS
```

1.5.2 COBOL Programs

To initialize API storage structures in COBOL programs, move blanks, zeros, or explicit values in the fields, depending on the field type.

Note: COBOL copybook field names are similar to the names used by the Assembler macros except:

- COBOL substitutes the underscore character (`_`) with a hyphen (`-`). For example, `AACTL_SHUTDOWN` appears as `AACTL-SHUTDOWN` and `ALELM_RQ` appears as `ALELM-RQ`.
- COBOL does not allow the use of the pound character (`#`). For example, `AACTL_#SELECTED` appears as `AACTL-SELECTED`.

The code below shows how to initialize the ECHACTL control structure:

```
INITIALIZE      AACTL-DATAAREA.
```

This code shows an example of initializing the request and response data portion of the ECHALELM copybook.

```
INITIALIZE      ALELM-RQ-DATAAREA.  
INITIALIZE      ALELM-RS-DATAAREA.
```

Note: The control, request, and response header portions of each copybook are initialized by COBOL value clauses. These values must not be changed.

1.6 Starting Up and Shutting Down the API Server

Your first API function call automatically starts the API server, which transfers data to and from Endeavor and your application program. Your last API function call must shut the API server down.

To shut down the API server, set the ECHACTL Shutdown field to 'Y'. Below is an example of COBOL code setting the AACTL-SHUTDOWN field to 'N':

```
MOVE 'N'          TO AACTL-SHUTDOWN.
```

1.7 Calling the API from an Assembler Program

The following Assembler code shows a sample API function call for a list environment request. The code:

1. Begins by defining the control structure through the ENHAACTL macro, and the list environment function call request and response structures through the ENHALENV macro.
2. Initializes the ENHAACTL control structure, sets the API server Shutdown field (AACTL_SHUTDOWN) to 'N', and sets the DD names of the message and list environment response files.
3. Initializes the list environment function call request and response structures.
4. Defines the search criteria for the function call; in this case, the ALENV_RQ fields are set to conduct a logical search in all environments and return after finding the first occurrence.
5. Loads the addresses of the ENHAACTL control structure, ALENV_RQ request structure, and ALENV_RS response structure in a parameter list stored in register 1.
6. Calls the API server using the ENA\$NDVR interface program.

```

                                ENHAACTL DSECT=NO
                                ENHALENV DSECT=NO
PARMLIST DC      3F'0'
API$INIT STG=AACTL,BLOCK=AACTL      Initialize the ctl block
MVC  AACTL_SHUTDOWN,C'N'           Do not shutdown API server
MVC  AACTL_MSG_DDN,=CL8'MSG3FILE'  Set Message DD name
MVC  AACTL_LIST_DDN,=CL8'EXT1ELM'  Set DD name for List Env
API$INIT STG=ALENV_RQ,BLOCK=ALENV_RQ Initialize the req block
API$INIT STG=ALENV_RS,BLOCK=ALENV_RS Initialize the rsp block
MVI  ALENV_RQ_PATH,C'L'           Set to Logical Search
MVI  ALENV_RQ_RETURN,C'F'         Set to return first hit
MVI  ALENV_RQ_SEARCH,C'A'        Set search to ALL
MVC  ALENV_RQ_ENV,=CL8'PRDENV'    Set the environ name
LA   R1,PARMLIST                 Set up the parm list
LA   R14,AACTL                   R1 -> parmlist
ST   R14,0(0,R1)                 0(,R1) = A(AACTL)
LA   R14,ALENV_RQ                4(,R1) = A(ALENV_RQ)
ST   R14,4(0,R1)                 8(,R1) = A(ALENV_RS)
LA   R14,ALENV_RS
ST   R14,8(0,R1)
OI   8(R1),X'80'                 TURN ON HIGHORDER BIT
L    R15,=V(ENA$NDVR)
BALR R14,R15                     CALL SERVER THRU ENA$NDVR

```

1.8 Calling the API from a COBOL Program

Copybooks are provided for all API structures. Below are examples of API function calls from a COBOL program. Note that the first parameter refers to a control structure, the second a request structure which is followed by a response structure:

Template for List and Extract Function Call

```
CALL EAC-ENDEVOR-APINAME USING AACTL ALELM-RQ ALELM-RS.
```

Template for Element Action Function Call

```
CALL EAC-ENDEVOR-APINAME USING AACTL AEGEN-RQ.
```

In the above examples, EAC-ENDEVOR-APINAME contains the name of the Endeavor API interface program which is set to the value of ENA\$NDVR.

In the first example, ALELM-RQ describes a list element request structure and the ALELM-RS describes the list element response structure. Through the request structure, Endeavor inventory data, pathing information, and return options are specified. After the API processes the list element request, it places the first response in the list element response structure, ALELM-RS. All data responses, including the first, are written to the file identified by the AACTL-LIST-DDN field. If this field is not specified, no responses are written to this file.

1.9 Checking API Return and Reason Codes

Once the API server processes an API function call, it is recommended to check the return and reason codes returned to the control structure. The return code indicates the severity of the error and the reason code indicates the cause of the error. For example, return code 04 combined with reason code 002 indicates a warning message resulting from not finding a requested stage. See -- Heading 'RETRET' unknown -- for a complete list of the return and reason codes you might expect to get.

To check the return and reason codes returned from an API function call, examine the AACTL_RTNCODE and AACTL_REASON fields in the control structure. You can also check the return code value in register 15. Your return code will be one of the following values:

Return Code	Description
00	I - Informational. Processing concluded normally. Message is issued for informational purposes only.
04	W - Warning. An error was encountered which was not serious enough to terminate processing.
08	C - Caution. An error was encountered which may prevent further processing.
12	E - Error. An error was encountered that terminated processing of the current action but allowed Endeavor to continue with the next action request.
16	S - Severe. A severe error was encountered that prevented Endeavor from completing the requested action. Processing will terminate immediately. This category includes internal, system, and I/O errors.
20	F - Fatal. No further processing is possible.

In addition, you should check the value of the ACCTL_HI_MSGID field, which contains one of the following values:

- The highest API message ID encountered while processing the API function call. The value of this field has the following format:

 APIxxyyyz

 where *xx* is the return code, *yyy* is the reason code, and *z* is the severity letter.
- The error message ID returned by Endeavor. This could be a source management error, inventory management error, and so on.

1.10 Writing Messages to the Message File

You must define a message file to write output messages to. The API server writes the messages to the message file DD name that you supply to the AACTL_MSG_DDN field. Endeavor messages, such as the Execution Report and Source and Inventory Management errors (if any), are recorded to this data set. This is NOT a required field, but it is highly recommended that a file name be provided. If one is not specified, a default DD name of APIMSGS is used. It is the responsibility of the user to allocate this data set prior to executing an API function call. For example, if your JCL contains the following DD statement, assign MSG3FILE to the AACTL_MSG_DDN field:

```
//MSG3FILE DD DSN=&&MSG3FILE,DISP=(NEW,PASS),  
//          UNIT=SYSDA,SPACE=(TRK,(5,5)),  
//          DCB=(RECFM=FB,LRECL=133,BLKSIZE=13300)
```

The API server manages this file. It is not necessary to define, open, or close this file in your application program.

1.11 Writing Responses to a Response File

To write the API responses to a file, define a response file that has a variable block record format and logical record length of 2048 bytes. For example:

```
//EXT1ELM DD DSN=&&EXT1ELM,DISP=(NEW,PASS),  
//          UNIT=SYSDA,SPACE=(TRK,(5,5)),  
//          DCB=(RECFM=VB,LRECL=2048,BLKSIZE=22800)
```

The file DCB must be: RECFM=VB,LRECL=2048 (calculate the minimum record size by adding 4 to the size of the response structure for the function you are requesting. In most cases, 2048 is large enough).

The API server writes the responses to the response file DD name that you supply to the AACTL_LIST_DD field. The API server manages this file. It is not necessary to define, open, or close this file in your application program unless you want to read it as input after the API server has finished writing all the responses. If you are only performing a query and do not need a file of responses, leave this field blank. The total count of records selected is included in the Execution Report and the AACTL_#SELECTED field. For example, if your JCL contains the DD statement above, assign EXT1ELM to the AACTL_LIST_DD field.

Note: If you request the list component/where-used function with the build SCL option, the file DCB must be: RECFM=FB,LRECL=80.

1.12 Sample Applications

To help you get started using the API, source code for sample COBOL and Assembler programs is provided in the `iprfx.igual.SOURCE` data set. Each of these programs issues function calls to the API.

- COBOL program, `CCIDRPT1`, produces a list of elements based on user input and creates a CCID cross-reference report. This program is distributed as a source module.
- Assembler program, `ENHAAPGM`, issues a list environment function call to list all the environments defined in the logical map and writes the output to a file. This program is distributed as a source module.
- Assembler program, `ENHAEPGM`, executes each of the element action function calls and writes the responses to a response file. This program is distributed as a source module.
- Assembler program, `ENHAPLST`, illustrates each of the list package function calls and writes the responses to a response file. This program is distributed as a source module.
- Assembler program, `ENHAPUPD`, illustrates each of the update type package action function calls and writes the responses to a response file. This program is distributed as a source module.
- Assembler program, `ENTBJAPI`, executes different inventory list function calls based on input. With this program, you can get familiar with the various inventory list function calls and the output that each function call generates. This program is distributed as a load module only.

Refer to Appendix A, “Sample API Programs” on page A-1 for additional information and the JCL required to execute these programs.

1.13 Documentation Overview

This manual is part of a comprehensive documentation set that fully describes the features and functions of Endeavor and explains how to perform everyday tasks. For a complete list of Endeavor manuals, see the PDF Table of Contents file in the PDF directory, or the Bookmanager Bookshelf file in the Books directory.

The following section describes product conventions.

1.14 Name Masking

A name mask allows you to specify all names, or all names beginning with a particular string, to be considered when performing an action.

Name masks are valid on:

- Element names
- System, subsystem, and type names within FROM clauses
- Report syntax
- ISPF panels
- API requests
- Package IDs

Name masks are not valid on:

- Environment names
- Element names in the following situations:
 - When entering a LEVel in a statement
 - When using the MEMber clause with a particular action
 - When building a package

1.14.1 Usage

There are three ways to mask names: by using the wildcard character (*), by using the placeholder character (%), and by using both together.

The wildcard (*) can be used in one of two ways to specify external file names:

- When coded as the only character of a search string, Endeavor returns all members of the search field. For example, if you coded the statement `ADD ELEMENT *`, all elements would be added.
- When coded as the last character of a search string, Endeavor returns all members of the search field beginning with the characters in the search string preceding the wildcard. For example:
 - The statement `ADD ELEMENT UPD*` would add all elements beginning with "UPD", such as `UPDATED` or `UPDATE`.
 - `PKG*` would return all package IDs beginning with `PKG`.

Note: You cannot use more than one wildcard in a string. The statement `ADD ELEMENT U*PD*` would result in an error.

The placeholder (%), which represents any one character in a string, can also be used in one of two ways:

- When coded as the last character in a string, Endeavor returns all members of the search field, beginning with the characters in the search string preceding the placeholder, but which have no more characters than were coded in the search string.
 - If you coded the statement `ADD ELEMENT UPD%`, only those elements with four-character-long names beginning with "UPD" (UPD1 or UPDA, for example) would be added.
 - `PKG%` returns PKGS, PKGB, PKGC, and so on.
- It is also possible to use the placeholder multiple times in a single search string. The statement `ADD ELEMENT U%PD%` would return all elements with five-character-long names that have U as the first character, and PD third and fourth.

The wildcard and the placeholder can be used together, provided that the wildcard appears only at the end of the search string and is used only once. For example:

- The statement `ADD ELEMENT U%D*`, which uses both the wildcard and the placeholder, would add elements with names of any length that have U as the first character, any one character as the second character, and D as the third character.
- `P%G*` returns PKGABCD, POGS, PIGGY, PPG1234NDVR, and so on.

Chapter 2. API Function Calls

2.1 Overview

This chapter describes the fields contained in the control, request, request extension and response structures for each Endeavor API function call. The information is grouped by Element actions, Package actions and Inventory List actions. The Assembler macro layouts and COBOL copybook layouts are delivered in the iprfx.iqual.SOURCE installation library.

Before issuing an API function call, you must initialize the control, request, request extension and response structures. You must also populate some of the data areas in the control and request structures. For instructions on initializing structures, see 1.5, “Initializing API Structures” on page 1-12.

2.2 Control Structure

The control structure contains the field that allows you to shutdown the API server as well as define the response and message files. It also contains the return code, the reason code and the selection count for the function call that you have performed.

The first call to the API server automatically starts it. The server remains open until the SHUTDOWN field is set to Y.

The control structure also contains the information about the output files used by the API on each function call. Each function call opens and closes a response file which is defined in this structure. If two or more function calls use the same response file, only the response to the last function call is maintained. Data from the prior function call is overwritten.

Assembler: ENHAACTL

COBOL: ECHAACTL

2.2.1 AACTL Control Structure Fields

The following table contains a list of the fields contained in the control structure and the options available to you.

Field	Length	Description
AACTL_MSG_DDN	Character 8	Message file DD name. Endeavor messages, such as the Execution Report and Source and Inventory Management errors (if any), are recorded to this data set. This is NOT a required field, but it is highly recommended that a file name be provided. If one is not specified, a default DD name of APIMSGS is used. It is the responsibility of the user to allocate this data set prior to executing an API function call.

Field	Length	Description
AACTL_LIST_DDN	Character 8	<p>Response file DD name. Inventory list response records, extract element response records and package list response records are written to this data set. The layout of each record corresponds to the response structure. This is NOT a required field. It is ignored by API element and package update type action processing. You need to allocate this data set prior to executing an API request. The file DCB must be: RECFM=VB,LRECL=2048 (calculating the minimum record size by adding 4 to the size of the response structure for the function you are requesting. In most cases, 2048 works).</p> <p>Note: If you request the list component/where-used function with the build SCL option, the file DCB must be: RECFM=FB,LRECL=80.</p>
AACTL_HI_MSGID	Character 8	<p>The API updates this field with the highest message id encountered while processing the request.</p> <p>Format: APIxxyyz where: xx = return code yyy = reason code z = severity letter</p>
AACTL_SHUTDOWN	Character 1	<p>API server Startup/Shutdown flag. Set to N to invoke the API server. Set to Y to shutdown the server.</p>
AACTL_RTNCODE	Character 4	<p>Return code from processing the request. Updated by the API. See -- Heading 'RETRET' unknown -- for more information.</p>

Field	Length	Description
AACTL_REASON	Character 4	Reason code from processing the request. Updated by the API. See -- Heading 'RETRET' unknown -- for more information.
AACTL_#SELECTED	Character 8	<p>The number of responses that meet your selection criteria. If you are attempting to perform a query and are just interested in the total count, this field contains that value. This value is also displayed on the Execution report.</p> <p>For the element list function, it is possible for the #SELECTED count to be greater than the #RETURNED count. This can occur if the "from/to" extension records exist and you request them.</p> <p>Note: If you specify the CSECT option in the List Directory function request structure, the returned count (AACTL_#RETURNED) contains the number of members processed and the selected count (AACTL_#SELECTED) contains the total number of CSECTs found in the returned members.</p>

Field	Length	Description
AACTL_#RETURNED	Character 8	<p>The number of responses written to the response or list file. If you do not specify a response file DDN, the value of this field will be 1. This indicates one response structure was returned to the calling program. This value is also displayed on the Execution report. Element action function calls do not use this field.</p> <p>Note: If you specify the CSECT option in the List Directory function request structure, the returned count (AACTL_#RETURNED) contains the number of members processed and the selected count (AACTL_#SELECTED) contains the total number of CSECTs found in the returned members.</p>
AACTL_STOPRC	Binary (2 bytes)	<p>Maximum allowable processing return code. This value is used only when you specify wildcarding on the request, for example, add element ab*. If not specified, the default value of 16 is used. In cases where the location information is fully qualified, the field is ignored.</p> <p>Note: You can set the default to 12 in the Optional Features Table.</p>
AACTL_ERRMSG	Character 132	First most severe message.

Field	Length	Description
AACTL_CMDMSG_DD	Character 8	Package execution log message report file DD name. All execution messages created by the package execute action are written to this report file. This is not a required field. If a value is not specified, a default DD name of C1EXMSG is used. It is the responsibility of the user to allocate this data set prior to executing an API request.

2.3 API Function Calls

Endevor functions supported by the API can be categorized into four groups:

- **Element Extract**

Allows you to extract element and/or component source from the base and delta libraries. Also, you can extract summary, changes, and history information associated with element or component data.

- **Element Actions**

Allows you to perform one of the element actions, such as ADD, GENERATE, MOVE, or PRINT ELEMENT. These functions retrieve information from the MCF and base and delta libraries, and may update these files, depending on the function.

- **Package Actions**

Allows you to perform package actions. This includes APPROVE, BACKOUT, CAST, COMMIT, DEFINE, DELETE, DENY, EXECUTE, RESET or SUBMIT.

- **Inventory Query and List Functions**

Allows you to request inventory lists and perform queries. This includes elements, environments, packages, systems, subsystems, and all other inventory information stored in the MCF. You can also request directory lists and component/where-used lists.

2.3.1 Understanding Logical and Physical Mapping Requests

Many request structures ask you to specify whether the mapping path is logical or physical:

- Logical mapping refers to how the system administrator logically sets the route for the inventory.
- Physical mapping refers to a direct physical path.

Within an environment, inventory always goes from Stage 1 to Stage 2. But, when an administrator maps across to another environment, the administrator may choose Stage 1 or Stage 2 of that environment. If the administrator chooses Stage 1 of the second environment, Stage 1 and Stage 2 of that environment represent the route. If the administrator chooses Stage 2 of the second environment, Stage 1 of that environment does not become part of the logical map, but does remain part of the physical map.

The example below illustrates the difference. Suppose you have Environment A and Environment B, both with Stage 1 and Stage 2. Suppose further that the system administrator maps Environment A to Stage 2 of Environment B.

In a physical mapping, the inventory route always includes Stage 1 of Environment B even though the system administrator maps to Stage 2:

1. Environment A / Stage 1

2. Environment A / Stage 2
3. Environment B / Stage 1
4. Environment B / Stage 2

In a logical map, the inventory route bypasses Stage 1 of Environment B, and the inventory route becomes:

1. Environment A / Stage 1
2. Environment A / Stage 2
3. Environment B / Stage 2

2.4 Request Extension

Unlike the request structures, there is only one request extension call to support all the API functions. The API request extension defines the parameters necessary to process long name elements or HFS file structures. If you are not processing long name elements or HFS files, you can ignore the API request extension. If specified, it must be the third parameter in the calling sequence to the API interface program, ENA\$NDVR, following the function call.

Assembler: ENHAAREB

COBOL: ECHAAREB

2.4.1.1 AAREB_RQ Request Extension Structure Fields

The API request extension defines the parameters necessary to process long name elements or HFS file structures. When using the AAREB structure, you must specify all element and member information in the AAREB structure while the element, member, data set and DD name fields in the request structure must remain blank.

Field	Length	Description
AAREB_RQ_ELM	Character 255	Element name
	Character 1	Alignment
AAREB_RQ_ELM_THRU	Character 255	Through element name
	Character 1	Alignment
AAREB_RQ_ELM_TELM	Character 255	Target element name or through HFS file name
or AAREB_RQ_HFSF_THRU	Character 1	Alignment
AAREB_RQ_PATH	Character 768	HFS path name
AAREB_RQ_HFSF	Character 255	HFS file name
AAREB_RQ_RESERVE	Character 2	** Reserved field **
	Character 2	Alignment

2.5 Add Element Action

The add element action API function call allows you to add an element to Endeavor.

Assembler: ENHAEADD

COBOL: ECHAEADD

2.5.1 AEADD_RQ Request Structure Fields

Immediately following the header is the data area of the AEADD_RQ request structure. The request structure is where you set your selection criteria.

All request selection fields are explained in the following table. For default values, see the *SCL Reference Guide*.

Field	Length	Description
AEADD_RQ_ELM	Character 10	Element name
AEADD_RQ_ELM_THRU	Character 10	Through element name
AEADD_RQ_DDN	Character 8	File or DD name where member(s) reside. Either DDN or DSN must be specified, but not both.
AEADD_RQ_DSN	Character 44	Data set name where member(s) reside. Either DDN or DSN must be specified, but not both.
AEADD_RQ_MBR	Character 10	From PDS member name
AEADD_RQ_ENV	Character 8	Environment name
AEADD_RQ_SYSTEM	Character 8	System name
AEADD_RQ_SUBSYS	Character 8	Subsystem name
AEADD_RQ_TYPE	Character 8	Type name
AEADD_RQ_CCID	Character 12	Change control id
AEADD_RQ_COMM	Character 40	Comment
AEADD_RQ_NEWVER	Character 2	New version (1-99)
AEADD_RQ_UPDT	Character 1	Update if present (Y/N)
AEADD_RQ_DEL	Character 1	Delete input source (Y/N)
AEADD_RQ_OVESIGNO	Character 1	Override signout (Y/N)
AEADD_RQ_BYP_GEN	Character 1	Bypass generate processor (Y/N)
AEADD_RQ_PROGRO	Character 8	Processor group name

2.5 Add Element Action

Field	Length	Description
AEADD_RQ_RESERVE	Character 3	** Reserved field **

2.6 Delete Element Action

The delete element action API function call deletes an element from the specified inventory location.

Assembler: ENHAEDEL

COBOL: ECHAEDEL

2.6.1 AEDEL_RQ Request Structure Fields

Immediately following the header is the data area of the AEDEL_RQ request structure. The request structure is where you set your selection criteria.

All request selection fields are explained in the following table. For default values, see the *SCL Reference Guide*.

The where CCID clause is limited to eight CCID values and the where processor group clause is limited to eight processor group names.

Field	Length	Description
AEDEL_RQ_ELM	Character 10	Element name
AEDEL_RQ_ELM_THRU	Character 10	Through element name
AEDEL_RQ_ENV	Character 8	Environment name
AEDEL_RQ_SYSTEM	Character 8	System name
AEDEL_RQ_SUBSYS	Character 8	Subsystem name
AEDEL_RQ_TYPE	Character 8	Type name
AEDEL_RQ_STG_ID	Character 1	Stage id. Either stage id or stage number must be specified, but not both.
AEDEL_RQ_STG_NUM	Character 1	Stage number (1/2). Either stage id or stage number must be specified, but not both.
AEDEL_RQ_CCID	Character 12	Change control id
AEDEL_RQ_COMM	Character 40	Comment
AEDEL_RQ_ONLY_CMP	Character 1	Only delete components (Y/N)
AEDEL_RQ_OVESIGNO	Character 1	Override signout (Y/N)
AEDEL_RQ_RESERVE	Character 3	** Reserved field **

Field	Length	Description
AEDEL_RQ_WCCID_TYP	Character 1	Where CCID type (A/C/R). C is the default value. If activated, the CCID_WHERE_ALL optional feature changes the default to A.
AEDEL_RQ_WCCID	Character 12	1st where CCID value
	Character 84	2nd through 8th where CCID values
AEDEL_RQ_WPROGRO	Character 8	1st where processor group value
	Character 56	2nd through 8th processor group values

- The first part is the prefix which contains the location, inventory classification, the record number, and the record's data length.
- The second part contains the element or component source data record.

2.7 Extract Element and Component Data

The extract element and component data API function call extracts elements and component data that match the criteria specified in the AEELM_RQ request structure and places it in the response file defined in the control structure. AEELM_RS, the response structure, contains information about the location and inventory classification of the element, as well as the number of records returned by the function call and the record length of the longest record.

Through the API, you can retrieve element data in both unformatted and formatted display styles and component data in formatted display style. You can also retrieve change and history information for both element and component data.

Assembler: ENHAEELM

COBOL: ECHAEELM

2.7.1 Element and Component Extraction Types

The extract element and component data function call provides seven types of extractions. These are specified with the AEELM_RQ_FORMAT and the AEELM_RQ_RTYPE fields:

- The AEELM_RQ_RTYPE field determines whether element or component data is extracted.
- The AEELM_RQ_FORMAT field determines the format that the information will appear in the response file.

The seven extraction types that result in different output record layouts are:

Extract Type	Output Record Layout
Element extract	Unformatted
Element extract, Browse	Browse element record format
Element extract, Change	Change element record format
Element extract, History	History element record format
Component extract, Browse	Browse component record format
Component extract, Change	Change component record format
Component extract, History	History component record format

2.7.2 AEELM_RQ Request Structure Fields

Immediately following the header is the data area of the AEELM_RQ request structure. The request structure is where you set your selection criteria. All request selection fields are explained in the following table:

Field	Length	Description
AEELM_RQ_FORMAT	Character 1	Format type: Blank - No format, just extract element to response file; use only with RTYPE of 'E' (for element). B - Endeavor Browse format C - Endeavor Change format H - Endeavor History format
AEELM_RQ_RTYPE	Character 1	Extraction record type: E - Element C - Component
AEELM_RQ_ENV	Character 8	Environment name. This field cannot contain a wildcard character.
AEELM_RQ_SYSTEM	Character 8	System name. This field cannot contain a wildcard character.
AEELM_RQ_SUBSYS	Character 8	Subsystem name. This field cannot contain a wildcard character.
AEELM_RQ_TYPE	Character 8	Type name. This field cannot contain a wildcard character.
AEELM_RQ_ELM	Character 10	Element name. This field cannot contain a wildcard character.
AEELM_RQ_STG_ID	Character 1	Stage id. This field cannot contain a wildcard character.
AEELM_RQ_VERSION	Character 2	Version number. If not specified, current version is assumed. Optional.
AEELM_RQ_LEVEL	Character 2	Level number. If not specified, current level is assumed. Optional.

2.7.3 AEELM_RS Response Structure Fields

Immediately following its header is the data area of the AEELM_RS response structure. The information contained in the response structure is explained in the following tables.

Field	Length	Description
AEELM_RS_SITE	Character 1	Site id
AEELM_RS_ENV	Character 8	Environment name
AEELM_RS_SYSTEM	Character 8	System name
AEELM_RS_SUBSYS	Character 8	Subsystem name
AEELM_RS_ELM	Character 10	Element name
AEELM_RS_TYPE	Character 8	Type name
AEELM_RS_STG_ID	Character 1	Stage id
AEELM_RS_RECcnt	Zoned Char 8	Number of records written to the response file.
AEELM_RS_MAXLEN	Zoned Char 8	Maximum record length encountered.

You can use the AEELM_RS_RECcnt and AEELM_RS_MAXLEN fields to determine the number of records written to the output file and the maximum record length encountered. The API server also identifies the record number and the record length for each record it writes to the response file. For each record, it replaces:

- The AEELM_RS_RECcnt field with the AEELM_RS_REC# (or in COBOL AEELM-RS-RECNUM) field, which identifies the record number.
- The AEELM_RS_MAXLEN field with the AEELM_RS_RECLEN field, which contains the record length, excluding any headers.

2.7.4 Element Extract and Component Data Record Layouts

Each element record returned in this format has two parts:

- The first part is the prefix which contains the location, inventory classification, the record number, and the record length of the extracted element record.
- The second part of the record contains the request element data.

Each component record returned has two parts:

- The first part is the prefix which contains the location, inventory classification, record number, and the record length of the specified element.
- The second part of the record contains the requested component data, including information about the component level, processor, symbol and macros.

2.7.4.2 Element Extract, Browse Record Layout

The following example shows the first 26 records found in the Browse Display format. In this example, the records shown have been truncated for display purposes. To view a complete record, invoke Endeavor on-line and go to the Display Element menu. Select Browse Option against any element.

```

EELR0INT NDVRMVS BASE TEST$API ASMPGM 1000000100000079*****
EELR0INT NDVRMVS BASE TEST$API ASMPGM 1000000200000079*****
EELR0INT NDVRMVS BASE TEST$API ASMPGM 1000000300000079**
EELR0INT NDVRMVS BASE TEST$API ASMPGM 1000000400000079** ELEMENT BROWSE
EELR0INT NDVRMVS BASE TEST$API ASMPGM 1000000500000079**
EELR0INT NDVRMVS BASE TEST$API ASMPGM 1000000600000079** ENVIRONMENT: INT SYSTEM: NDVRMVS SUBSYS
EELR0INT NDVRMVS BASE TEST$API ASMPGM 1000000700000079** ELEMENT: TEST$API TYPE: ASMPGM STAGE: I
EELR0INT NDVRMVS BASE TEST$API ASMPGM 1000000800000079**
EELR0INT NDVRMVS BASE TEST$API ASMPGM 1000000900000079*****
EELR0INT NDVRMVS BASE TEST$API ASMPGM 1000001000000079*****
EELR0INT NDVRMVS BASE TEST$API ASMPGM 1000001100000079
EELR0INT NDVRMVS BASE TEST$API ASMPGM 1000001200000079----- SOURCE LEVEL INFORMATION-----
EELR0INT NDVRMVS BASE TEST$API ASMPGM 1000001300000079
EELR0INT NDVRMVS BASE TEST$API ASMPGM 1000001400000099VV.LL SYNC USER DATE TIME STMTS CCID COMM
EELR0INT NDVRMVS BASE TEST$API ASMPGM 1000001500000099-----
EELR0INT NDVRMVS BASE TEST$API ASMPGM 100000160000009901.00 POCBR01 30APR99 16:00 395 APU api
EELR0INT NDVRMVS BASE TEST$API ASMPGM 100000170000009901.01 OLEJU01 09JAN00 09:49 584 JOAPI api
EELR0INT NDVRMVS BASE TEST$API ASMPGM 100000180000009901.02 OLEJU01 09JAN00 10:39 395 JOAPI keep
EELR0INT NDVRMVS BASE TEST$API ASMPGM 100000190000009901.03 OLEJU01 10FEB00 13:19 584 JOAPI add o
EELR0INT NDVRMVS BASE TEST$API ASMPGM 1000002000000099GENERATED OLEJU01 10FEB00 13:19 JOAPI add o
EELR0INT NDVRMVS BASE TEST$API ASMPGM 1000002100000099
EELR0INT NDVRMVS BASE TEST$API ASMPGM 1000002200000088+00 TEST$API $MODNTRY LINKAGE=EXT,STACK=25000,MSGDD=SYS
EELR0INT NDVRMVS BASE TEST$API ASMPGM 1000002300000088+00 COPY $QIODS
EELR0INT NDVRMVS BASE TEST$API ASMPGM 1000002400000088+00 MAIN $FUNCSTG ,
EELR0INT NDVRMVS BASE TEST$API ASMPGM 1000002500000088+00 GLOBALS DS 0D
EELR0INT NDVRMVS BASE TEST$API ASMPGM 1000002600000088+00 WDBLWORD DS D
    
```

2.7.4.3 Element Extract, Change Record Layout

The following example shows the first 26 records found in the Change Display format. In this example, the records shown have been truncated for display purposes. To view a complete record, invoke Endeavor on-line and go to the Display Element menu. Select Change Option against any element.

```

EELR0INT NDVRMVS BASE TEST$API ASMPGM 1000000100000079*****
EELR0INT NDVRMVS BASE TEST$API ASMPGM 1000000200000079*****
EELR0INT NDVRMVS BASE TEST$API ASMPGM 1000000300000079**
EELR0INT NDVRMVS BASE TEST$API ASMPGM 1000000400000079** ELEMENT CHANGES
EELR0INT NDVRMVS BASE TEST$API ASMPGM 1000000500000079**
EELR0INT NDVRMVS BASE TEST$API ASMPGM 1000000600000079** ENVIRONMENT: INT SYSTEM: NDVRMVS SUBSYS
EELR0INT NDVRMVS BASE TEST$API ASMPGM 1000000700000079** ELEMENT: TEST$API TYPE: ASMPGM STAGE
EELR0INT NDVRMVS BASE TEST$API ASMPGM 1000000800000079**
EELR0INT NDVRMVS BASE TEST$API ASMPGM 1000000900000079*****
EELR0INT NDVRMVS BASE TEST$API ASMPGM 1000001000000079*****
EELR0INT NDVRMVS BASE TEST$API ASMPGM 1000001100000079
EELR0INT NDVRMVS BASE TEST$API ASMPGM 1000001200000079----- SOURCE LEVEL INFORMATION-----
EELR0INT NDVRMVS BASE TEST$API ASMPGM 1000001300000079
EELR0INT NDVRMVS BASE TEST$API ASMPGM 1000001400000099VV.LL SYNC USER DATE TIME STMTS CCID COM
EELR0INT NDVRMVS BASE TEST$API ASMPGM 1000001500000099-----
EELR0INT NDVRMVS BASE TEST$API ASMPGM 100000160000009901.00 POCBR01 30APR99 16:00 395 APU api
EELR0INT NDVRMVS BASE TEST$API ASMPGM 100000170000009901.01 OLEJU01 09JAN00 09:49 584 JOAPI api
EELR0INT NDVRMVS BASE TEST$API ASMPGM 100000180000009901.02 OLEJU01 09JAN00 10:39 395 JOAPI kee
EELR0INT NDVRMVS BASE TEST$API ASMPGM 100000190000009901.03 OLEJU01 10FEB00 13:19 584 JOAPI add
EELR0INT NDVRMVS BASE TEST$API ASMPGM 1000002000000099GENERATED OLEJU01 10FEB00 13:19 JOAPI add
EELR0INT NDVRMVS BASE TEST$API ASMPGM 1000002100000099
EELR0INT NDVRMVS BASE TEST$API ASMPGM 1000002200000088+03 WINDEX_ALTYP EQU 7 . . .
EELR0INT NDVRMVS BASE TEST$API ASMPGM 1000002300000088+03 WINDEX_ALPGR EQU 8 . . .
EELR0INT NDVRMVS BASE TEST$API ASMPGM 1000002400000088+03 WINDEX_ALDSN EQU 9 . . .
EELR0INT NDVRMVS BASE TEST$API ASMPGM 1000002500000088+03 WINDEX_ALAGR EQU 10 . . .
EELR0INT NDVRMVS BASE TEST$API ASMPGM 1000002600000088+03 WINDEX_ALAGJ EQU 11 . . .
    
```

2.7.4.4 Element Extract, History Record Layout

The following example shows the first 38 records found in the History Display format. In this example, the records shown have been truncated for display purposes. To view a complete record, invoke Endeavor on-line and go to the Display Element menu. Select the History Option against any element.

```

EELR0INT NDVRMVS BASE TEST$API ASMPGM 1000000100000079*****
EELR0INT NDVRMVS BASE TEST$API ASMPGM 1000000200000079*****
EELR0INT NDVRMVS BASE TEST$API ASMPGM 1000000300000079**
EELR0INT NDVRMVS BASE TEST$API ASMPGM 1000000400000079** ELEMENT BROWSE
EELR0INT NDVRMVS BASE TEST$API ASMPGM 1000000500000079**
EELR0INT NDVRMVS BASE TEST$API ASMPGM 1000000600000079** ENVIRONMENT: INT SYSTEM:NDVRMVS SUBSYS
EELR0INT NDVRMVS BASE TEST$API ASMPGM 1000000700000079** ELEMENT: TEST$API TYPE: ASMPGM STAGE I
EELR0INT NDVRMVS BASE TEST$API ASMPGM 1000000800000079**
EELR0INT NDVRMVS BASE TEST$API ASMPGM 1000000900000079*****
EELR0INT NDVRMVS BASE TEST$API ASMPGM 1000001000000079*****
EELR0INT NDVRMVS BASE TEST$API ASMPGM 1000001100000079
EELR0INT NDVRMVS BASE TEST$API ASMPGM 1000001200000079----- SOURCE LEVEL INFORMATION-----
EELR0INT NDVRMVS BASE TEST$API ASMPGM 1000001300000079
EELR0INT NDVRMVS BASE TEST$API ASMPGM 1000001400000099VV.LL SYNC USER DATE TIME STMTS CCID COMM
EELR0INT NDVRMVS BASE TEST$API ASMPGM 1000001500000099-----
EELR0INT NDVRMVS BASE TEST$API ASMPGM 100000160000009901.00 POCBR01 30APR99 16:00 395 APU api
EELR0INT NDVRMVS BASE TEST$API ASMPGM 100000170000009901.01 OLEJU01 09JAN00 09:49 584 JOAPI api
EELR0INT NDVRMVS BASE TEST$API ASMPGM 100000180000009901.02 OLEJU01 09JAN00 10:39 395 JOAPI keep
EELR0INT NDVRMVS BASE TEST$API ASMPGM 100000190000009901.03 OLEJU01 10FEB00 13:19 584 JOAPI add o
EELR0INT NDVRMVS BASE TEST$API ASMPGM 1000002000000099GENERATED OLEJU01 10FEB00 13:19 JOAPI add o
EELR0INT NDVRMVS BASE TEST$API ASMPGM 1000002100000099
EELR0INT NDVRMVS BASE TEST$API ASMPGM 1000002200000088+00 TEST$API $MODNTRY LINKAGE=EXT,STACK=25000,MSGDD=SYS
EELR0INT NDVRMVS BASE TEST$API ASMPGM 1000002300000088+00 COPY $QIODS
EELR0INT NDVRMVS BASE TEST$API ASMPGM 1000002400000088+00 MAIN $FUNCSTG ,
EELR0INT NDVRMVS BASE TEST$API ASMPGM 1000002500000088+00 GLOBALS DS 0D
EELR0INT NDVRMVS BASE TEST$API ASMPGM 1000002600000088+00 WDBLWORD DS D
EELR0INT NDVRMVS BASE TEST$API ASMPGM 1000002700000088+00 MAINRSLT DS CL8
EELR0INT NDVRMVS BASE TEST$API ASMPGM 1000002800000088+00 MAIN@QIO DS A ADDRESS OF
EELR0INT NDVRMVS BASE TEST$API ASMPGM 1000002900000088+00 MAINMISS DS F MISMATCH C
EELR0INT NDVRMVS BASE TEST$API ASMPGM 1000003000000088+00 WINDEX DS F LAST FUNCT
EELR0INT NDVRMVS BASE TEST$API ASMPGM 1000003100000088+00 WINDEX_ALENV EQU 1 . . .
EELR0INT NDVRMVS BASE TEST$API ASMPGM 1000003200000088+00 WINDEX_ALSTG EQU 2 . . .
EELR0INT NDVRMVS BASE TEST$API ASMPGM 1000003300000088+00 WINDEX_ALSYS EQU 3 . . .
EELR0INT NDVRMVS BASE TEST$API ASMPGM 1000003400000088+00 WINDEX_ALSBS EQU 4 . . .
EELR0INT NDVRMVS BASE TEST$API ASMPGM 1000003500000088+00 WINDEX_ALSIT EQU 5 . . .
EELR0INT NDVRMVS BASE TEST$API ASMPGM 1000003600000088+00 WINDEX_ALELM EQU 6 . . .
EELR0INT NDVRMVS BASE TEST$API ASMPGM 1000003700000088%+03 WINDEX_ALTYP EQU 7 . . .
EELR0INT NDVRMVS BASE TEST$API ASMPGM 1000003800000088%+03 WINDEX_ALPGR EQU 8 . . .
    
```

2.7.4.5 Component Extract, Browse Record Layout

The following example shows the first 42 records found in the Browse Display format. In this example, the records shown have been truncated for display purposes. To view a complete record, invoke Endeavor on-line and go to the Display Element menu. Select the Browse Component Option against any element.

```

EELR0INT NDVRMVS BASE TEST$API ASMPGM 1000000100000079*****
EELR0INT NDVRMVS BASE TEST$API ASMPGM 10000000200000079*****
EELR0INT NDVRMVS BASE TEST$API ASMPGM 10000000300000079**
EELR0INT NDVRMVS BASE TEST$API ASMPGM 10000000400000079** COMPONENT BROWSE
EELR0INT NDVRMVS BASE TEST$API ASMPGM 10000000500000079**
EELR0INT NDVRMVS BASE TEST$API ASMPGM 10000000600000079** ENVIRONMENT: INT SYSTEM:NDVRMVS SUBSYS
EELR0INT NDVRMVS BASE TEST$API ASMPGM 10000000700000079** ELEMENT: TEST$API TYPE: ASMPGM STAGE
EELR0INT NDVRMVS BASE TEST$API ASMPGM 10000000800000079**
EELR0INT NDVRMVS BASE TEST$API ASMPGM 10000000900000079*****
EELR0INT NDVRMVS BASE TEST$API ASMPGM 10000001000000079*****
EELR0INT NDVRMVS BASE TEST$API ASMPGM 10000001100000079
EELR0INT NDVRMVS BASE TEST$API ASMPGM 10000001200000079----- COMPONENT LEVEL INFORMATION-----
EELR0INT NDVRMVS BASE TEST$API ASMPGM 10000001300000079
EELR0INT NDVRMVS BASE TEST$API ASMPGM 10000001400000099VV.VV.LL SYNC USER DATE TIME STMTS CCID COMM
EELR0INT NDVRMVS BASE TEST$API ASMPGM 10000001500000099-----
EELR0INT NDVRMVS BASE TEST$API ASMPGM 1000000160000009901.00 POCBR01 30APR99 16:00 71 APU api
EELR0INT NDVRMVS BASE TEST$API ASMPGM 1000000170000009901.01 OLEJU01 09JAN00 09:49 66 JOAPI api
EELR0INT NDVRMVS BASE TEST$API ASMPGM 1000000180000009901.02 OLEJU01 09JAN00 10:39 72 JOAPI keep
EELR0INT NDVRMVS BASE TEST$API ASMPGM 1000000190000009901.03 OLEJU01 09JAN00 10:41 72 JOAPI keep
EELR0INT NDVRMVS BASE TEST$API ASMPGM 1000000200000009901.04 OLEJU01 21JAN00 12:38 74 JOAPI api
EELR0INT NDVRMVS BASE TEST$API ASMPGM 1000000210000009901.05 OLEJU01 10FEB00 13:19 80 JOAPI add
EELR0INT NDVRMVS BASE TEST$API ASMPGM 10000002200000099
EELR0INT NDVRMVS BASE TEST$API ASMPGM 10000002300000000
EELR0INT NDVRMVS BASE TEST$API ASMPGM 10000002400000000
EELR0INT NDVRMVS BASE TEST$API ASMPGM 10000002500000079----- ELEMENT INFORMATION -----
EELR0INT NDVRMVS BASE TEST$API ASMPGM 10000002600000000
EELR0INT NDVRMVS BASE TEST$API ASMPGM 100000027000000101 VV.LL DATE TIME SYSTEM SUBSYS ELEMENT TY
EELR0INT NDVRMVS BASE TEST$API ASMPGM 100000028000000100%+05 01.03 10FEB00 13:19 NDVRMVS BASE TEST$API AS
EELR0INT NDVRMVS BASE TEST$API ASMPGM 10000002900000000
EELR0INT NDVRMVS BASE TEST$API ASMPGM 10000003000000000
EELR0INT NDVRMVS BASE TEST$API ASMPGM 10000003100000079----- PROCESSOR INFORMATION -----
EELR0INT NDVRMVS BASE TEST$API ASMPGM 10000003200000000
EELR0INT NDVRMVS BASE TEST$API ASMPGM 100000033000000101 VV.LL DATE TIME SYSTEM SUBSYS ELEMENT TY
EELR0INT NDVRMVS BASE TEST$API ASMPGM 100000034000000100+04 01.09 05JUL97 17:34 LGNTLCL PROCESS GASM PR
EELR0INT NDVRMVS BASE TEST$API ASMPGM 10000003500000000
EELR0INT NDVRMVS BASE TEST$API ASMPGM 10000003600000000
EELR0INT NDVRMVS BASE TEST$API ASMPGM 10000003700000078----- SYMBOL INFORMATION -----
EELR0INT NDVRMVS BASE TEST$API ASMPGM 10000003800000000
EELR0INT NDVRMVS BASE TEST$API ASMPGM 100000039000000135 DEFINED SYMBOL VALUE
EELR0INT NDVRMVS BASE TEST$API ASMPGM 100000040000000135+03 PROCESSOR AUTH 0
EELR0INT NDVRMVS BASE TEST$API ASMPGM 100000041000000135+00 PROCESSOR LET NOLET
EELR0INT NDVRMVS BASE TEST$API ASMPGM 100000042000000135+00 PROCESSOR LINK YES

```

2.7.4.6 Component Extract, Change Record Layout

The following example shows the first 41 records found in the Change Display format. In this example, the records shown have been truncated for display purposes. To view a complete record, invoke Endeavor on-line and go to the Display Element menu. Select the Change Component Option against any element.

```

EELR0INT NDVRMVS BASE TEST$API ASMPGM 10000000100000079*****
EELR0INT NDVRMVS BASE TEST$API ASMPGM 10000000200000079*****
EELR0INT NDVRMVS BASE TEST$API ASMPGM 10000000300000079**
EELR0INT NDVRMVS BASE TEST$API ASMPGM 10000000400000079** COMPONENT CHANGES
EELR0INT NDVRMVS BASE TEST$API ASMPGM 10000000500000079**
EELR0INT NDVRMVS BASE TEST$API ASMPGM 10000000600000079** ENVIRONMENT: INT SYSTEM:NDVRMVS SUBSYS
EELR0INT NDVRMVS BASE TEST$API ASMPGM 10000000700000079** ELEMENT: TEST$API TYPE: ASMPGM STAGE
EELR0INT NDVRMVS BASE TEST$API ASMPGM 10000000800000079**
EELR0INT NDVRMVS BASE TEST$API ASMPGM 10000000900000079*****
EELR0INT NDVRMVS BASE TEST$API ASMPGM 10000001000000079*****
EELR0INT NDVRMVS BASE TEST$API ASMPGM 10000001100000079
EELR0INT NDVRMVS BASE TEST$API ASMPGM 10000001200000079----- COMPONENT LEVEL INFORMATION-----
EELR0INT NDVRMVS BASE TEST$API ASMPGM 10000001300000079
EELR0INT NDVRMVS BASE TEST$API ASMPGM 10000001400000099VV.LL SYNC USER DATE TIME STMTS CCID COMM
EELR0INT NDVRMVS BASE TEST$API ASMPGM 10000001500000099-----
EELR0INT NDVRMVS BASE TEST$API ASMPGM 1000000160000009901.00 POCBR01 30APR99 16:00 71 APU api
EELR0INT NDVRMVS BASE TEST$API ASMPGM 1000000170000009901.01 OLEJU01 09JAN00 09:49 66 JOAPI api
EELR0INT NDVRMVS BASE TEST$API ASMPGM 1000000180000009901.02 OLEJU01 09JAN00 10:39 72 JOAPI keep
EELR0INT NDVRMVS BASE TEST$API ASMPGM 1000000190000009901.03 OLEJU01 09JAN00 10:41 72 JOAPI keep
EELR0INT NDVRMVS BASE TEST$API ASMPGM 1000000200000009901.04 OLEJU01 21JAN00 12:38 74 JOAPI api
EELR0INT NDVRMVS BASE TEST$API ASMPGM 1000000210000009901.05 OLEJU01 10FEB00 13:19 80 JOAPI add
EELR0INT NDVRMVS BASE TEST$API ASMPGM 10000002200000099
EELR0INT NDVRMVS BASE TEST$API ASMPGM 10000002300000000
EELR0INT NDVRMVS BASE TEST$API ASMPGM 10000002400000000
EELR0INT NDVRMVS BASE TEST$API ASMPGM 10000002500000079----- ELEMENT INFORMATION -----
EELR0INT NDVRMVS BASE TEST$API ASMPGM 10000002600000000
EELR0INT NDVRMVS BASE TEST$API ASMPGM 10000002700000101 VV.LL DATE TIME SYSTEM SUBSYS ELEMENT TY
EELR0INT NDVRMVS BASE TEST$API ASMPGM 10000002800000100+05 01.03 10FEB00 13:19 NDVRMVS BASE TEST$API AS
EELR0INT NDVRMVS BASE TEST$API ASMPGM 10000002900000100+03-05 01.02 09JAN00 10:39 NDVRMVS BASE TEST$API AS
EELR0INT NDVRMVS BASE TEST$API ASMPGM 10000003000000000
EELR0INT NDVRMVS BASE TEST$API ASMPGM 10000003100000000
EELR0INT NDVRMVS BASE TEST$API ASMPGM 10000003200000079----- INPUT COMPONENTS -----
EELR0INT NDVRMVS BASE TEST$API ASMPGM 10000003300000000
EELR0INT NDVRMVS BASE TEST$API ASMPGM 10000003400000000
EELR0INT NDVRMVS BASE TEST$API ASMPGM 10000003500000086STEP: ASSEM DD=SYSLIB VOL=NDVR01 DSN=BST.INTMVSS1.MAC
EELR0INT NDVRMVS BASE TEST$API ASMPGM 10000003600000000
EELR0INT NDVRMVS BASE TEST$API ASMPGM 10000003700000096 MEMBER VV.LL DATE TIME SYSTEM SUBSYS E
EELR0INT NDVRMVS BASE TEST$API ASMPGM 10000003800000095+05 ENHAEELM 01.00 10JAN00 14:17 NDVRMVS BASE E
EELR0INT NDVRMVS BASE TEST$API ASMPGM 10000003900000095+05 ENHALAGJ 01.00 10JAN00 14:18 NDVRMVS BASE E
EELR0INT NDVRMVS BASE TEST$API ASMPGM 10000004000000095+05 ENHALAGR 01.00 10JAN00 14:18 NDVRMVS BASE E
EELR0INT NDVRMVS BASE TEST$API ASMPGM 10000004100000095+05 ENHALDSN 01.00 10JAN00 14:18 NDVRMVS BASE E
    
```

2.7.4.7 Component Extract, History Record Layout

The History display lists all of the components and events related to an element. The following example shows the first 41 records found in the History display format. In this example, the records shown have been truncated for display purposes. To view a complete record, invoke Endeavor on-line and go to the Display Element menu. Select the History Component Option against any element.

```

EELR0INT NDVRMVS BASE TEST$API ASMPGM 10000000100000079*****
EELR0INT NDVRMVS BASE TEST$API ASMPGM 10000000200000079*****
EELR0INT NDVRMVS BASE TEST$API ASMPGM 10000000300000079**
EELR0INT NDVRMVS BASE TEST$API ASMPGM 10000000400000079** COMPONENT HISTORY
EELR0INT NDVRMVS BASE TEST$API ASMPGM 10000000500000079**
EELR0INT NDVRMVS BASE TEST$API ASMPGM 10000000600000079** ENVIRONMENT: INT SYSTEM:NDVRMVS SUBSYS
EELR0INT NDVRMVS BASE TEST$API ASMPGM 10000000700000079** ELEMENT: TEST$API TYPE: ASMPGM STAGE
EELR0INT NDVRMVS BASE TEST$API ASMPGM 10000000800000079**
EELR0INT NDVRMVS BASE TEST$API ASMPGM 10000000900000079*****
EELR0INT NDVRMVS BASE TEST$API ASMPGM 10000001000000079*****
EELR0INT NDVRMVS BASE TEST$API ASMPGM 10000001100000079
EELR0INT NDVRMVS BASE TEST$API ASMPGM 10000001200000079----- COMPONENT LEVEL INFORMATION-----
EELR0INT NDVRMVS BASE TEST$API ASMPGM 10000001300000079
EELR0INT NDVRMVS BASE TEST$API ASMPGM 10000001400000099VV.LL SYNC USER DATE TIME STMTS CCID COMM
EELR0INT NDVRMVS BASE TEST$API ASMPGM 10000001500000099-----
EELR0INT NDVRMVS BASE TEST$API ASMPGM 1000000160000009901.00 POCBR01 30APR99 16:00 71 APU api
EELR0INT NDVRMVS BASE TEST$API ASMPGM 1000000170000009901.01 OLEJU01 09JAN00 09:49 66 JOAPI api
EELR0INT NDVRMVS BASE TEST$API ASMPGM 1000000180000009901.02 OLEJU01 09JAN00 10:39 72 JOAPI keep
EELR0INT NDVRMVS BASE TEST$API ASMPGM 1000000190000009901.03 OLEJU01 09JAN00 10:41 72 JOAPI keep
EELR0INT NDVRMVS BASE TEST$API ASMPGM 1000000200000009901.04 OLEJU01 21JAN00 12:38 74 JOAPI api
EELR0INT NDVRMVS BASE TEST$API ASMPGM 1000000210000009901.05 OLEJU01 10FEB00 13:19 80 JOAPI add
EELR0INT NDVRMVS BASE TEST$API ASMPGM 10000002200000099
EELR0INT NDVRMVS BASE TEST$API ASMPGM 10000002300000000
EELR0INT NDVRMVS BASE TEST$API ASMPGM 10000002400000000
EELR0INT NDVRMVS BASE TEST$API ASMPGM 10000002500000079----- ELEMENT INFORMATION -----
EELR0INT NDVRMVS BASE TEST$API ASMPGM 10000002600000000
EELR0INT NDVRMVS BASE TEST$API ASMPGM 100000027000000101 VV.LL DATE TIME SYSTEM SUBSYS ELEMENT TY
EELR0INT NDVRMVS BASE TEST$API ASMPGM 100000028000000100%+05 01.03 10FEB00 13:19 NDVRMVS BASE TEST$API AS
EELR0INT NDVRMVS BASE TEST$API ASMPGM 100000029000000100%+03-05 01.02 09JAN00 10:39 NDVRMVS BASE TEST$API AS
EELR0INT NDVRMVS BASE TEST$API ASMPGM 100000030000000100%+02-03 01.02 09JAN00 10:39 NDVRMVS BASE TEST$API AS
EELR0INT NDVRMVS BASE TEST$API ASMPGM 100000031000000100%+01-02 01.01 09JAN00 09:49 NDVRMVS BASE TEST$API AS
EELR0INT NDVRMVS BASE TEST$API ASMPGM 100000032000000100%+00-01 01.00 30APR99 16:00 NDVRMVS BASE TEST$API AS
EELR0INT NDVRMVS BASE TEST$API ASMPGM 10000003300000000
EELR0INT NDVRMVS BASE TEST$API ASMPGM 10000003400000000
EELR0INT NDVRMVS BASE TEST$API ASMPGM 10000003500000079----- PROCESSOR INFORMATION -----
EELR0INT NDVRMVS BASE TEST$API ASMPGM 10000003600000000
EELR0INT NDVRMVS BASE TEST$API ASMPGM 100000037000000101 VV.LL DATE TIME SYSTEM SUBSYS ELEMENT TY
EELR0INT NDVRMVS BASE TEST$API ASMPGM 100000038000000100%+04 01.09 05JUL97 17:34 LGNTLCL PROCESS GASMPR
EELR0INT NDVRMVS BASE TEST$API ASMPGM 100000039000000100%+03-04 01.03 16AUG98 11:26 LGNTLCL PROCESS GASMPR
EELR0INT NDVRMVS BASE TEST$API ASMPGM 100000040000000100%+00-03 01.09 05JUL97 17:34 LGNTLCL PROCESS GASMPR
EELR0INT NDVRMVS BASE TEST$API ASMPGM 10000004100000000

```

2.8 Generate Element Action

The generate element action API function call executes the generate processor for the current level of an element.

Assembler: ENHAEGEN

COBOL: ECHAEGEN

2.8.1 AEGEN_RQ Request Structure Fields

Immediately following the header is the data area of the AEGEN_RQ request structure. The request structure is where you set your selection criteria.

All request selection fields are explained in the following table. For default values, see the *SCL Reference Guide*.

The where CCID clause is limited to eight CCID values and the where processor group clause is limited to eight processor group names.

Field	Length	Description
AEGEN_RQ_ELM	Character 10	Element name
AEGEN_RQ_ELM_THRU	Character 10	Through element name
AEGEN_RQ_ENV	Character 8	Environment name
AEGEN_RQ_SYSTEM	Character 8	System name
AEGEN_RQ_SUBSYS	Character 8	Subsystem name
AEGEN_RQ_TYPE	Character 8	Type name
AEGEN_RQ_STG_ID	Character 1	Stage id. Either stage id or stage number must be specified, but not both.
AEGEN_RQ_STG_NUM	Character 1	Stage number (1/2). Either stage id or stage number must be specified, but not both.
AEGEN_RQ_CCID	Character 12	Change control id
AEGEN_RQ_COMM	Character 40	Comment
AEGEN_RQ_COPYBACK	Character 1	Copy back element (Y/N)
AEGEN_RQ_SEARCH	Character 1	Search map (Y/N). Y is the default
AEGEN_RQ_OVESIGNO	Character 1	Override signout (Y/N)
AEGEN_RQ_PROGRO	Character 8	Processor group name

Field	Length	Description
AEGEN_RQ_RESERVE	Character 3	** Reserved field **
AEGEN_RQ_WCCID_TYP	Character 1	Where CCID type (A/C/R). C is the default value. If activated, the CCID_WHERE_ALL optional feature changes the default to A.
AEGEN_RQ_WCCID	Character 12	1st where CCID value
	Character 84	2nd through 8th where CCID values
AEGEN_RQ_WPROGRO	Character 8	1st where processor group value
	Character 56	2nd through 8th processor group values

2.9 List Element

The list element API function call allows you to produce a list of elements. Depending on the response file DDN (AACTL_LIST_DDN) field value, the first of the last response structure is placed in your defined response area, ALELM_RS. Refer to 2.2, “Control Structure” on page 2-3 for details. This allows you to check the response quickly if you are looking for a specific element. The API also writes all responses generated by your request to an external data set if you specified a file output DD name in the control structure.

Assembler: ENHALELM

COBOL: ECHALELM

2.9.1 ALELM_RQ Request Structure Fields

Immediately following the header is the data area of the ALELM_RQ request structure. The request structure is where you set your selection criteria. All request selection fields are explained in the following table.

Field	Length	Description
ALELM_RQ_ENV	Character 8	Environment name. This field cannot contain a wildcard character.
ALELM_RQ_SYSTEM	Character 8	System name. This field can contain a wildcard character.
ALELM_RQ_SUBSYS	Character 8	Subsystem name. This field can contain a wildcard character.
ALELM_RQ_TYPE	Character 8	Type name. This field can contain a wildcard character.
ALELM_RQ_ELM	Character 10	Element name. This field can contain a wildcard character.
ALELM_RQ_STG_ID	Character 1	Stage id as defined in the C1DEFLTS table. You can code either stage id or stage number but not both. This field can contain a wildcard character.
ALELM_RQ_STG_NUM	Character 1	Stage number (1/2). You can code either stage id or stage number but not both.

Field	Length	Description
ALELM_RQ_TOENV	Character 8	Ending Environment name. Used in conjunction with "B"etween or "R"ange SEARCH options. A wildcard character is not allowed.
ALELM_RQ_TOSTG_ID	Character 1	Ending Stage id. Used in conjunction with the "B"etween or "R"ange SEARCH options. A wildcard character is not allowed.
ALELM_RQ_TOELM	Character 10	To Element name. If specified, this field can contain a wildcard. Optional.
ALELM_RQ_ELM_THRU	Character 10	Through Element name.

Searching: If you specify 'B', or 'R' for the ALELM_RQ_SEARCH search argument, you must specify the To Environment in the ALELM_RQ_TOENV field and the To Stage Id in the ALELM_RQ_TOSTG_ID field. You cannot use a wildcard. In addition, these fields will also be ignored unless you use them with the 'B' or 'R' search fields.

If you specify the ALELM_RQ_TOELM field, To Element Name, the element name must be greater than the ALELM_RQ_ELM field value, Element Name.

2.9.1.1 Information About Action Options

Field	Length	Description
ALELM_RQ_PATH	Character 1	Mapping path: L - for Logical P - for Physical
ALELM_RQ_RETURN	Character 1	F - for return only the first record that satisfies the request. A - for return all records that satisfy the request.

Field	Length	Description
ALELM_RQ_SEARCH	Character 1	<p>Mapping argument:</p> <p>A - for Search All the way up the map.</p> <p>B - for Search Between the two specified environments and stages.</p> <p>N - for No Search.</p> <p>E - for Search next specified environment/stage then up the map.</p> <p>R - for Search the Range, between and including the specified environments and stages.</p>
ALELM_RQ_BDATA	Character 1	<p>Only return the base element data.</p> <p>Y - Only return the base data. If this option is enabled, use the ALELB_RS structure defined in ENHALELM to map the response data fields.</p> <p>N - Return all element master data.</p>
ALELM_RQ_FDSN	Character 1	<p>Return the 'from' dataset-member/path-file data as a response record (ALELM_RS_RECTYP=F).</p> <p>Y - Return this data.</p> <p>N - Do not return this data.</p> <p>If you enable this option, use the ALELX_RS structure defined in ENHALELM to map the response data fields.</p>

Field	Length	Description
ALELM_RQ_TDSN	Character 1	<p>Return the 'target' dataset-member/path-file data as a response record (ALELM_RS_RECTYP=T).</p> <p>Y - Return this data.</p> <p>N - Do not return this data.</p> <p>If you enable this option, use the ALELM_RS structure defined in ENHALELM to map the response data fields.</p>

2.9.1.2 Information About Selection Criteria

Field	Length	Description
ALELM_RQ_WLPROC_FAIL	Character 1	Selection criteria. Where last processor failed (Y/N).
ALELM_RQ_WLPROC_TYP	Character 1	<p>Selection criteria. Where last processor type is:</p> <p>D - Delete Processor</p> <p>G - Generate Processor</p> <p>M - Move Processor</p>
ALELM_RQ_WGEN_FDATE	Character 7	<p>Selection criteria. Where generate from date. Format is DDMMYY (31JAN01).</p> <p>To select based on an exact date and time, code the from and through date and time with the identical values.</p> <p>If you specify the from information and not the through information, the API selects all records starting at the from date/time and later.</p>
ALELM_RQ_WGEN_FTIME	Character 5	Selection criteria. Where generate from time. Format is HH:MM (23:59).

Field	Length	Description
ALELM_RQ_WGEN_TDATE	Character 7	<p>Selection criteria. Where generate through date. Format is DDMMYY (31JAN01).</p> <p>To select based on an exact date and time, code the from and through date and time with the identical values.</p> <p>If you specify the through information and not the from information, the API selects all records starting at the through date/time and earlier.</p>
ALELM_RQ_WGEN_TTIME	Character 5	<p>Selection criteria. Where generate through time. Format is HH:MM (24:59).</p>
ALELM_RQ_WCCID_TYP	Character 1	<p>Where CCID type (A/C/G/L/R).</p> <p>A - Last action</p> <p>G - Generate</p> <p>L - Last level</p> <p>R - Retrieve</p> <p>C - Current CCID. Compares the WCCID value(s) against the last action, generate and last level CCID values. If any match is found, the record is selected.</p> <p>If this field is specified, at least one WCCID value must be specified. If this field is not specified, the WCCID field(s) are ignored.</p>
ALELM_RQ_WCCID	Character 12	1st where change control id value
ALELM_RQ_WCCID2_8	Character 84	2nd through 8th where ccid values
ALELM_RQ_WPROGRO	Character 8	1st where processor group value
ALELM_RQ_WPROGRO2_8	Character 56	2nd through 8th where processor group values
	Character 4	** Reserved field **

Note: The ALELM_RQ_FLAG field is now ALELM_RQ_BDATA and ALELM_RQ_TO_ELM is now ALELM_RQ_ELM_THRU. You should use the new names in new applications. The API, however, still supports the old field names. A value of 'B' or 'Y' activates the 'base fields only' feature.

2.9.2 List Element Response Structures

There are three types of list element response structures:

- **Master or full** contains all the fields located in the master control file record about an element. Depending on the value you specify in the ALELM_RQ_BDATA field in the request structure, you will receive all the element fields or just the Base fields.
- **Base** contains what are considered to be the more important MCF record fields. It is a subset of the master response structure.
- **From and target extension records** contain the path and file names when processing long name elements. The list element response block supports long path and HFS file names for *from* and *target* locations. When enabled, the ALELM_RQ_FDSN and ALELM_RQ_TDSN fields inform the API to return this data as response records. When activated, the API writes the *from* and/or *retrieve to* path and file data as response records. These records are referred to as extension records. If you request this data, but no extension records exist, the flag is ignored.

The API can create one, two, or three List Element response records. The element master always exists. The from element extension record, and the retrieve to extension record only exist if the data does not fit into the master record. If the path is less than 45 characters and the file name is less than 11 characters, extension records do not exist. In this case the DSN and MBR fields in the master record contain the actual values.

From extension record only exists under the following conditions:

- The from path name is greater than 44 characters, or
- The from file name is greater than 10 characters.

If either condition is met:

- The ALELM_RS_FR_DSN field contains a 1 in position 1, and
- The ALELM_RS_FR_MBR field is blank.

Retrieve to extension record only exists under the following conditions:

- The retrieve to path name is greater than 44 characters, or
- The from file name is greater than 10 characters.

If either condition is met:

- The ALELM_RS_RET_DSN field contains a 1 in position 1, and
- The ALELM_RS_RET_MBR field is blank.

Example: The following example shows all three records, the master record, the from record and the to record.

```
.M.....LELSM0I40      NDVRMVS BUCFR02 EQZMNUF7 HFSTYPE I40STG1 110001*OPROC*20
0D000500DCDEDFCFF44444DCEDDEE4CECCDF4CDEDECF44CCEEEDC4CFFEECF4FFFFFF5DDDDC5FF
3400410235324094000000545945202436902058945467008623875094023710110001C67963C20
-----
.....LELSF../u/users/endeavor/..testbsretrieved
03000500DCDEC016A6AA89A68988A99600A8AA8A98A988A88
010041023532601141425921554556910F352322953995554
-----
.....LELST../u/users/endeavor/..testbsretrieved
03000500DCDEE016A6AA89A68988A99600A8AA8A98A988A88
010041023532301141425921554556910F352322953995554
```

The RECTYP value is F for the from record, or T for the to record. The from record is identified by the F in LELSF; the to record is identified by the T in LELST. Preceding the path and HFS file name is a halfword containing the length of the value. In this example x'0011' (17) is the length of path and x'000F' (15) is the length of the HFS file name.

In this example, because the element master contains a value of 1 in position 1 of the 44 character ALELM_RS_FR_DSN field, a second response record, a "from" element extension record, is created by the API. Because the element master contains a value of 1 in position 1 of the 44 character ALELM_RS_RET_DSN field, a third response record, an element "retrieve to" extension record, is created by the API.

Notes:

1. The retrieve data is blank until a retrieve action is performed against this element. Coding a value of "Y" in the ALELM_RQ_FDSN and/or ALELM_RQ_TDSN fields is meaningless unless extension records exist. The ALELM_RS extension response block is meaningless unless extension records exist.
2. Regarding your request type for the the response structure, always specify the base response structure, ALELM_RS, as the last parameter in the call to the API (ENA\$NDVR).

2.9.3 ALELM_RS Response Structure Fields

Immediately following its header is the data area of the ALELM_RS response structure. The information contained in the response structure is explained in the following tables.

Field	Length	Description
ALELM_RS_RECTYP	Character 1	There are four types of response record. M - Master record B - Base data only F - From dataset-member/path-file data T - Target dataset-member/path-file data The values specified on the BDATA, FDSN and TDSN fields in the ALELM_RQ block determine the types of records returned.
ALELM_RS_SITE	Character 1	Site id
ALELM_RS_ENV	Character 8	Environment name
ALELM_RS_SYSTEM	Character 8	System name
ALELM_RS_SUBSYS	Character 8	Subsystem name
ALELM_RS_ELEMENT	Character 10	Element name
ALELM_RS_TYPE	Character 8	Type name
ALELM_RS_STG_NAME	Character 8	Stage name
ALELM_RS_STG_ID	Character 1	Stage id
ALELM_RS_STG_NUM	Character 1	Stage number (1/2)
ALELM_RS_STG_REL	Zoned Char 4	Relative mapped stage number
ALELM_RS_PROCGRP	Character 8	Processor group name
ALELM_RS_UPD_DATE	Zoned Char 8	Record update date YYYYMMDD
ALELM_RS_UPD_TIME	Zoned Char 8	Record update time HHMMSSTT
ALELM_RS_SIGNOUT	Character 8	Signout user id
ALELM_RS_ELM_VV	Zoned Char 2	Current element version number
ALELM_RS_ELM_LL	Zoned Char 2	Current element level number
ALELM_RS_CMP_VV	Character 2	Current component version number
ALELM_RS_CMP_LL	Character 2	Current component level number

2.9.3.1 Information About the Last Action

Field	Length	Description
ALELM_RS_LMOD_NAME	Character 8	Last element - modifying action
ALELM_RS_LACT_NAME	Character 8	Last action
ALELM_RS_LACT_RC	Character 5	Endevor return code
ALELM_RS_LACT_DATE	Zoned Char 8	Date YYYYMMDD
ALELM_RS_LACT_TIME	Zoned Char 8	Time HHMMSSTT
ALELM_RS_LACT_USER	Character 8	User id
ALELM_RS_LACT_CCID	Character 12	CCID
ALELM_RS_LACT_COMMENT	Character 40	Comment

2.9.3.2 Information About the Element Base

Field	Length	Description
ALELM_RS_EBAS_NAME	Character 10	Element base member name
ALELM_RS_EBAS_DATE	Zoned Char 8	Element base date YYYYMMDD
ALELM_RS_EBAS_TIME	Zoned Char 8	Element base time HHMMSSTT
ALELM_RS_EBAS_TOTL	Zoned Char 5	Number of statements in base
ALELM_RS_EBAS_LVL	Zoned Char 2	Base level number
ALELM_RS_EBAS_FLG1	Character 1	Y - if Element base is compressed
ALELM_RS_EBAS_USER	Character 8	User id associated with base
ALELM_RS_EBAS_COMMENT	Character 40	Comment associated with base

2.9.3.3 Information About the Element Delta (Last Level)

Field	Length	Description
ALELM_RS_EDLT_NAME	Character 8	Element delta member name
ALELM_RS_EDLT_DATE	Zoned Char 8	Element last level date YYYYMMDD

Field	Length	Description
ALELM_RS_EDLT_TIME	Zoned Char 8	Element last level time HHMMSSTT
ALELM_RS_EDLT_TOTL	Zoned Char 5	Number of statements in last level
ALELM_RS_EDLT_USER	Character 8	User id associated with last level
ALELM_RS_EDLT_CCID	Character 12	CCID associated with last level
ALELM_RS_EDLT_ COMMENT	Character 40	Comment associated with last level
ALELM_RS_EDLT_INS	Zoned Char 5	Number of inserts in last level
ALELM_RS_EDLT_DEL	Zoned Char 5	Number of deletes in last level
ALELM_RS_EDLT_FMT	Character 1	F - Forward delta format R - Reverse delta format

2.9.3.4 Information About the Component List Base

Field	Length	Description
ALELM_RS_XBAS_NAME	Character 8	Component base member name
ALELM_RS_XBAS_DATE	Zoned Char 8	Component base date YYYYMMDD
ALELM_RS_XBAS_TIME	Zoned Char 8	Component base time HHMMSSTT
ALELM_RS_XBAS_TOTL	Zoned Char 8	Number of statements in base
ALELM_RS_XBAS_LVL	Zoned Char 2	Base level number

2.9.3.5 Information About the Component List Delta

Field	Length	Description
ALELM_RS_XDLT_NAME	Character 8	Component delta member name
ALELM_RS_XDLT_DATE	Zoned Char 8	Component last level date YYYYMMDD
ALELM_RS_XDLT_TIME	Zoned Char 8	Component last level time HHMMSSTT

Field	Length	Description
ALELM_RS_XDLT_TOTL	Zoned Char 8	Number of statements in last level
ALELM_RS_XDLT_INS	Zoned Char 8	Number of inserts in last level
ALELM_RS_XDLT_DEL	Zoned Char 8	Number of deletes in last level
ALELM_RS_XDLT_FMT	Character 1	F - Forward delta format R - Reverse delta format
ALELM_RS_XDLT_MON	Character 1	M - Component list built by monitor
ALELM_RS_XDLT_CPY	Character 1	C - Component list copied or restored
ALELM_RS_XDLT_DLTA	Character 1	D - Component list base stored in the delta library

2.9.3.6 Information About the Last Element Move

Field	Length	Description
ALELM_RS_MOV_DATE	Zoned Char 8	Element Move date YYYYMMDD
ALELM_RS_MOV_TIME	Zoned Char 8	Element Move time HHMMSSTT
ALELM_RS_MOV_USER	Character 8	User id associated with the move

2.9.3.7 Information About the Last Add or Update Data Set

Field	Length	Description
ALELM_RS_FR_DSN	Character 44	Add/Update from data set name
ALELM_RS_FR_MBR	Character 10	Add/Update from member name

2.9.3.8 Information About the Element Processor Execution

Field	Length	Description
ALELM_RS_PROC_FLG1	Character 1	0 - No processor executed 1 - Last processor was Generate 2 - Last processor was Move 3 - Last processor was Delete 4 - Last processor was unknown
ALELM_RS_PROC_FLG2	Character 1	F - Last processor failed
ALELM_RS_PROC_DATE	Zoned Char 8	Element last processor date YYYYMMDD
ALELM_RS_PROC_TIME	Zoned Char 8	Element last processor time HHMMSSTT
ALELM_RS_GEN_DATE	Zoned Char 8	Element last generate date YYYYMMDD
ALELM_RS_GEN_TIME	Zoned Char 8	Element last generate time HHMMSSTT
ALELM_RS_GEN_USER	Character 8	User id associated with last generate
ALELM_RS_GEN_CCID	Character 12	CCID associated with last generate
ALELM_RS_GEN_ COMMENT	Character 40	Comment associated with last generate
ALELM_RS_PROC_FLG	Character 8	'*FAILED*' - if last processed failed execution
ALELM_RS_LASTPROC	Character 8	Name of the last processor executed
ALELM_RS_PROC_RC	Character 5	Processor return code

2.9.3.9 Information About the Last Element Retrieve

Field	Length	Description
ALELM_RS_RET_DATE	Zoned Char 8	Last Retrieve date YYYYMMDD
ALELM_RS_RET_TIME	Zoned Char 8	Last Retrieve time HHMMSSTT
ALELM_RS_RET_USER	Character 8	User id associated with last retrieve

Field	Length	Description
ALELM_RS_RET_CCID	Character 12	CCID associated with last retrieve
ALELM_RS_RET_COMM	Character 40	Comment associated with last retrieve
ALELM_RS_RET_DSN	Character 44	Retrieve-To data set name
ALELM_RS_RET_MBR	Character 10	Retrieve-To member name

2.9.3.10 Information About the Package Last Executed Against the Element Source

Field	Length	Description
ALELM_RS_SPKG_DATE	Zoned Char 8	Date YYYYMMDD
ALELM_RS_SPKG_TIME	Zoned Char 8	Time HHMMSSSTT
ALELM_RS_SPKG_ID	Character 16	Package id

2.9.3.11 Information About the Package Last Executed Against the Element Outputs

Field	Length	Description
ALELM_RS_OPKG_DATE	Zoned Char 8	Date YYYYMMDD
ALELM_RS_OPKG_TIME	Zoned Char 8	Time HHMMSSSTT
ALELM_RS_OPKG_ID	Character 16	Package id

2.9.3.12 Information About the Last "FROM" Endeavor Location

Field	Length	Description
ALELM_RS_FROM_SITE	Character 1	Site id
ALELM_RS_FROM_ENV	Character 8	Environment name
ALELM_RS_FROM_SYS	Character 8	System name
ALELM_RS_FROM_SBS	Character 8	Subsystem name
ALELM_RS_FROM_ELM	Character 10	Element name
ALELM_RS_FROM_TYPE	Character 8	Type name
ALELM_RS_FROM_STG#	Character 1	Stage number
ALELM_RS_FROM_FACT	Character 8	Action that updated "From" location

Field	Length	Description
ALELM_RS_FROM_VV	Character 2	Version number
ALELM_RS_FROM_LL	Character 2	Level number

2.9.3.13 Other Fields

Field	Length	Description
ALELM_RS_FMID	Zoned Char 5	Record created release id

2.9.3.14 Information About the Package for which the Element is Locked

Field	Length	Description
ALELM_LPKG_DATE	Zoned Char 8	DATE YYYYMMDD
ALELM_LPKG_TIME	Zoned Char 8	TIME HHMSSTT
ALELM_LPKG_ID	Character 16	Package id
	Character 1	** Reserved field **
ALELM_RS_ELMNAMEL	Binary (2 bytes)	Length of element name. Possible values are 1-255.
ALELM_RS_ELMNAME	Character 255	Full element name. If the element name exceeds 10 characters in length, this field contains the full element name and ALELM_RS_ELM contains a generated short name. If the name is not less than 10 characters, ELMNAME and ELM are equal.
	Character 1	** Reserved field **

2.9.3.15 ALELB_RS Response Structure Fields

There are three types of list element response structures:

- **Master or full** contains all the fields located in the master control file record about an element. Depending on the value you specify in the ALELM_RQ_BDATA field in the request structure, you will receive all the element fields or just the Base fields.
- **Base** contains what are considered to be the more important MCF record fields. It is a subset of the master response structure.
- **From and target** extension records contain the path and file names when processing long name elements.

Extension records only exist when the from/target path name exceeds 44 characters and/or the from/target HFS file name is greater than 10 characters. When written, the RECTYP value is 'F' or 'T' and they follow the master or base response record.

'Y' - Only returns the base data. If this option is enabled, use the ALELB_RS structure defined in ENHALELM to map the response data fields.

'N' - Returns all element master data.

Field	Length	Description
ALELB_RS_RECTYP	Character 1	Record type: 'B' base data 'M' master record 'F' long name from extension record 'T' long name to extension record
ALELB_RS_SITE	Character 1	Site id
ALELB_RS_ENV	Character 8	Environment name
ALELB_RS_SYSTEM	Character 8	System name
ALELB_RS_SUBSYS	Character 8	Subsystem name
ALELB_RS_ELEMENT	Character 10	Element name
ALELB_RS_TYPE	Character 8	Type name
ALELB_RS_STG_NAME	Character 8	Stage name
ALELB_RS_STG_ID	Character 1	Stage id
ALELB_RS_STG_NUM	Character 1	Stage number
ALELB_RS_STG_REL	Zoned Char 4	Relative stage number
ALELB_RS_PROCGRP	Character 8	Processor group
ALELB_RS_UPD_DATE	Zoned Char 8	Record update date 'YYYYMMDD'
ALELB_RS_UPD_TIME	Zoned Char 8	Record update time 'HHMMSSTT'
ALELB_RS_SIGNOUT	Character 8	Signout userid
ALELB_RS_ELM_VV	Zoned Char 2	Current element version number
ALELB_RS_ELM_LL	Zoned Char 2	Current element level number
ALELB_RS_CMP_VV	Character 2	Current component version number
ALELB_RS_CMP_LL	Character 2	Current component level number
ALELB_RS_EDLT_DATE	Zoned Char 8	Element last level date 'YYYYMMDD'

Field	Length	Description
ALELB_RS_EDLE_TIME	Zoned Char 8	Element last level time 'HHMMSSTT'
ALELB_RS_ELMNAMEL	Binary (2 bytes)	Length of element name. Possible values are 1-255.
ALELB_RS_ELMNAME	Character 255	Actual element name 'L' for length of element name
	Character 1	Alignment

If you enable the ALELM_RQ_FDSN or ALELM_RQ_TDSN options, use the ALELX_RS structure defined in ENHALELM to map the response data fields.

2.9.3.16 ALELX_RS Response Structure Fields

Field	Length	Description
ALELX_RS_RECTYP	Character 1	Record type: 'B' base data 'M' master record 'F' long name from extension record 'T' long name to extension record
ALELX_RS_EXTDATA	Character 1027	Extension record data. The format of this area consists of a two-byte binary path length, followed by a variable length path name, followed by a two-byte binary file name length, followed by a variable length file name. <ul style="list-style-type: none"> ■ FROM/TO PATH name length. Field length is 2 bytes (halfword). ■ FROM/TO PATH name. Field length is 45-768 bytes (variable). ■ FROM/TO FILE name length. Field length is 2 bytes (halfword). ■ FROM/TO FILE name. Field length is 11-255 bytes (variable).

2.10 Move Element Action

The move element action API function call moves elements between inventory locations along a map.

Assembler: ENHAEMOV

COBOL: ECHAEMOV

2.10.1 AEMOV_RQ Request Structure Fields

Immediately following the header is the data area of the AEMOV_RQ request structure. The request structure is where you set your selection criteria.

All request selection fields are explained in the following table. For default values, see the *SCL Reference Guide*.

The where CCID clause is limited to eight CCID values and the where processor group clause is limited to eight processor group names.

Field	Length	Description
AEMOV_RQ_ELM	Character 10	Element name
AEMOV_RQ_ELM_THRU	Character 10	Through element name
AEMOV_RQ_ENV	Character 8	Environment name
AEMOV_RQ_SYSTEM	Character 8	System name
AEMOV_RQ_SUBSYS	Character 8	Subsystem name
AEMOV_RQ_TYPE	Character 8	Type name
AEMOV_RQ_STG_ID	Character 1	Stage id. Either stage id or stage number must be specified, but not both.
AEMOV_RQ_STG_NUM	Character 1	Stage number (1/2). Either stage id or stage number must be specified, but not both.
AEMOV_RQ_CCID	Character 12	Change control id
AEMOV_RQ_COMM	Character 40	Comment
AEMOV_RQ_SYN	Character 1	Synchronize (Y/N)
AEMOV_RQ_WIT_HIS	Character 1	Move with history (Y/N)
AEMOV_RQ_BYP_DEL	Character 1	Bypass element delete (Y/N)

Field	Length	Description
AEMOV_RQ_SIGNIN	Character 1	Sign-in element (Y/N). Y is the default
AEMOV_RQ_RETAIN_SIGNO	Character 1	Retain sign-out (Y/N)
AEMOV_RQ_SIGNO_TO	Character 8	Sign-out element to userid
AEMOV_RQ_JUMP	Character 1	Move across environment
AEMOV_RQ_RESERVE	Character 3	** Reserved field **
AEMOV_RQ_WCCID_TYP	Character 1	Where CCID type (A/C/R). C is the default value. If activated, the CCID_WHERE_ALL optional feature changes the default to A.
AEMOV_RQ_WCCID	Character 12	1st where CCID value
	Character 84	2nd through 8th where CCID values
AEMOV_RQ_WPROGRO	Character 8	1st where processor group value
	Character 56	2nd through 8th processor group values

2.11 Print Element Action

The print element action API function call prints selected information about an element or library member, depending on the data entered in the FROM clause. You can print from Endeavor or selected output libraries (for example, a PDS).

Assembler: ENHAEPRE

COBOL: ECHAEPRE

2.11.1 AEPRE_RQ Request Structure Fields

Immediately following the header is the data area of the AEPRE_RQ request structure. The request structure is where you set your selection criteria.

All request selection fields are explained in the following table. For default values, see the *SCL Reference Guide*.

The where CCID clause is limited to eight CCID values and the where processor group clause is limited to eight processor group names. The stage id field is supported by the API, but not by SCL.

Field	Length	Description
AEPRE_RQ_ELM	Character 10	Element name
AEPRE_RQ_ELM_THRU	Character 10	Through element name
AEPRE_RQ_VERSION	Character 2	Element version (1-99)
AEPRE_RQ_LEVEL	Character 2	Element level (0-99)
AEPRE_RQ_ENV	Character 8	Environment name
AEPRE_RQ_SYSTEM	Character 8	System name
AEPRE_RQ_SUBSYS	Character 8	Subsystem name
AEPRE_RQ_TYPE	Character 8	Type name
AEPRE_RQ_STG_ID	Character 1	Stage id. Either stage id or stage number must be specified, but not both.
AEPRE_RQ_STG_NUM	Character 1	Stage number (1/2). Either stage id or stage number must be specified, but not both.
AEPRE_RQ_DDN	Character 8	File or DD name. CIPRINT is the default
AEPRE_RQ_NOCC	Character 1	Suppress headings (Y/N)

Field	Length	Description
AEPRE_RQ_COMP	Character 1	Request is for component data (Y/N)
AEPRE_RQ_PRT_OPT	Character 1	Type of print requested (B/C/H/M/S). B is the default. The M option is not allowed if the request is for component data
AEPRE_RQ_SEARCH	Character 1	Search map (Y/N). Y is the default
AEPRE_RQ_RESERVE	Character 3	** Reserved field **
AEPRE_RQ_WCCID_TYP	Character 1	Where CCID type (A/C/R). C is the default value. If activated, the CCID_WHERE_ALL optional feature changes the default to A.
AEPRE_RQ_WCCID	Character 12	1st where CCID value
	Character 84	2nd through 8th where CCID values
AEPRE_RQ_WPROGRO	Character 8	1st where processor group value
	Character 56	2nd through 8th processor group values

2.12 Print Member Element Action

The print member element action API function call prints selected information about the member you specify. You can print from Endeavor or from selected output libraries (for example, a PDS).

Assembler: ENHAEPRM

COBOL: ECHAEPRM

2.12.1 AEPRM_RQ Request Structure Fields

Immediately following the header is the data area of the AEPRM_RQ request structure. The request structure is where you set your selection criteria.

All request selection fields are explained in the following table. For default values, see the *SCL Reference Guide*.

Field	Length	Description
AEPRM_RQ_MBR	Character 10	Member name
AEPRM_RQ_MBR_THRU	Character 10	Through member name
AEPRM_RQ_DDN	Character 8	Source file or DD name where member(s) reside. Either DDN or DSN must be specified, but not both.
AEPRM_RQ_DSN	Character 44	Source data set name where member(s) reside. Either DDN or DSN must be specified, but not both.
AEPRM_RQ_TDDN	Character 8	Target file or DD name. C1PRINT is the default.

2.13 Retrieve Element Action

The retrieve element action API function call copies an element to a user data set.

Assembler: ENHAERET

COBOL: ECHAERET

2.13.1 AERET_RQ Request Structure Fields

Immediately following the header is the data area of the AERET_RQ request structure. The request structure is where you set your selection criteria.

All request selection fields are explained in the following table. For default values, see the *SCL Reference Guide*.

The where CCID clause is limited to eight CCID values and the where processor group clause is limited to eight processor group names.

Field	Length	Description
AERET_RQ_ELM	Character 10	Element name
AERET_RQ_ELM_THRU	Character 10	Through element name
AERET_RQ_VERSION	Character 2	Element version (1-99)
AERET_RQ_LEVEL	Character 2	Element level (0-99)
AERET_RQ_ENV	Character 8	Environment name
AERET_RQ_SYSTEM	Character 8	System name
AERET_RQ_SUBSYS	Character 8	Subsystem name
AERET_RQ_TYPE	Character 8	Type name
AERET_RQ_STG_ID	Character 1	Stage id. Either stage id or stage number must be specified, but not both.
AERET_RQ_STG_NUM	Character 1	Stage number (1/2). Either stage id or stage number must be specified, but not both.
AERET_RQ_FILE	Character 8	Target file or DD name. Either DDN or DSN must be specified, but not both
AERET_RQ_DSN	Character 44	Target data set name. Either DDN or DSN must be specified, but not both

2.13 Retrieve Element Action

Field	Length	Description
AERET_RQ_MBR	Character 10	Target PDS member name must be blank when through element name is specified.
AERET_RQ_CCID	Character 12	Change control id
AERET_RQ_COMM	Character 40	Comment
AERET_RQ_REPLACE	Character 1	Replace member (Y/N)
AERET_RQ_NO_SIGNO	Character 1	No signout (Y/N)
AERET_RQ_EXPAND	Character 1	Expand includes (Y/N)
AERET_RQ_OVESIGNO	Character 1	Override signout (Y/N)
AERET_RQ_SEARCH	Character 1	Search map (Y/N). Y is the default.
AERET_RQ_RESERVE	Character 3	** Reserved field **
AERET_RQ_WCCID_ TYP	Character 1	Where CCID type (A/C/R). C is the default value. If activated, the CCID_WHERE_ALL optional feature changes the default to A.
AERET_RQ_WCCID	Character 12	1st where CCID value
	Character 84	2nd through 8th where CCID values
AERET_RQ_WPROGRO	Character 8	1st where processor group value
	Character 56	2nd through 8th processor group values

2.14 Signin Element Action

The signin element action API function call removes a user signout associated with an element. It also enables you to sign out or reassign an element to another user.

Assembler: ENHAESIG

COBOL: ECHAESIG

2.14.1 AESIG_RQ Request Structure Fields

Immediately following the header is the data area of the AESIG_RQ request structure. The request structure is where you set your selection criteria.

All request selection fields are explained in the following table. For default values, see the *SCL Reference Guide*.

The where CCID clause is limited to eight CCID values and the where processor group clause is limited to eight processor group names.

Field	Length	Description
AESIG_RQ_ELM	Character 10	Element name
AESIG_RQ_ELM_THRU	Character 10	Through element name
AESIG_RQ_ENV	Character 8	Environment name
AESIG_RQ_SYSTEM	Character 8	System name
AESIG_RQ_SUBSYS	Character 8	Subsystem name
AESIG_RQ_TYPE	Character 8	Type name
AESIG_RQ_STG_ID	Character 1	Stage id
AESIG_RQ_STG_NUM	Character 1	Stage number (1/2)
AESIG_RQ_OVESIGNO	Character 1	Override signout (Y/N)
AESIG_RQ_SIGNO_TO	Character 8	Signout to userid
AESIG_RQ_SEARCH	Character 1	Search map (Y/N). Y is the default.
AESIG_RQ_RESERVE	Character 3	** Reserved field **
AESIG_RQ_WCCID_TYP	Character 1	Where CCID type (A/C/R). C is the default value. If activated, the CCID_WHERE_ALL optional feature changes the default to A.
AESIG_RQ_WCCID	Character 12	1st where CCID value

Field	Length	Description
	Character 84	2nd through 8th where CCID values
AESIG_RQ_WPROGRO	Character 8	1st where processor group value
	Character 56	2nd through 8th processor group values

2.15 Transfer Element Action

The transfer element action API function call transfers an element from one location to another.

Note: Only the Endeavor to Endeavor type transfer action is supported.

Assembler: ENHAETRA

COBOL: ECHAETRA

2.15.1 AETRA_RQ Request Structure Fields

Immediately following the header is the data area of the AETRA_RQ request structure. The request structure is where you set your selection criteria.

All request selection fields are explained in the following table. For default values, see the *SCL Reference Guide*.

The where CCID clause is limited to eight CCID values and the where processor group clause is limited to eight processor group names.

Field	Length	Description
AETRA_RQ_ELM	Character 10	From element name
AETRA_RQ_ELM_THRU	Character 10	Through element name
AETRA_RQ_VERSION	Character 2	From element version (1-99). Not allowed if thru element is specified.
AETRA_RQ_LEVEL	Character 2	From element level (0-99). Not allowed if thru element is specified.
AETRA_RQ_TELM	Character 10	Target element name. Not allowed if thru element is specified.
AETRA_RQ_ENV	Character 8	From environment name
AETRA_RQ_SYSTEM	Character 8	From system name
AETRA_RQ_SUBSYS	Character 8	From subsystem name
AETRA_RQ_TYPE	Character 8	From type name
AETRA_RQ_STG_ID	Character 1	From stage id. Either stage id or stage number must be specified, but not both.
AETRA_RQ_STG_NUM	Character 1	From stage number (1/2). Either stage id or stage number must be specified, but not both.

Field	Length	Description
AETRA_RQ_TENV	Character 8	Target environment name
AETRA_RQ_TSYSTEM	Character 8	Target system name
AETRA_RQ_TSUBSYS	Character 8	Target subsystem name
AETRA_RQ_TTYPE	Character 8	Target type name
AETRA_RQ_TSTG_ID	Character 1	Target stage id. Either target stage id or target stage number must be specified, but not both.
AETRA_RQ_TSTG_NUM	Character 1	Target stage number (1/2). Either target stage id or target stage number must be specified, but not both.
AETRA_RQ_CCID	Character 12	Change control id
AETRA_RQ_COMM	Character 40	Comment
AETRA_RQ_NEWVER	Character 2	New target element version
AETRA_RQ_IGN_GFAIL	Character 1	Ignore generate fail
AETRA_RQ_BYP_GENPRO	Character 1	Bypass generate processor
AETRA_RQ_PROGRO	Character 8	Processor group name
AETRA_RQ_OVESIGNO	Character 1	Override signout (Y/N)
AETRA_RQ_BYP_ELMDEL	Character 1	Bypass element delete (Y/N)
AETRA_RQ_BYP_DELPRO	Character 1	Bypass delete processor (Y/N)
AETRA_RQ_WIT_HIS	Character 1	Move with history (Y/N)
AETRA_RQ_SYN	Character 1	Synchronize (Y/N)
AETRA_RQ_SIGNIN	Character 1	Sign-in element (Y/N)
AETRA_RQ_RETA_SIGNO	Character 1	Retain sign-out (Y/N)
AETRA_RQ_SIGNO_TO	Character 8	Sign-out element to this user
AETRA_RQ_RESERVE	Character 3	** Reserved field **
AETRA_RQ_WCCID_TYP	Character 1	Where CCID type (A/C/R). C is the default value. If activated, the CCID_WHERE_ALL optional feature changes the default to A.
AETRA_RQ_WCCID	Character 12	1st where CCID value

Field	Length	Description
	Character 84	2nd through 8th where CCID values
AETRA_RQ_WPROGRO	Character 8	1st where processor group value
	Character 56	2nd through 8th processor group values

2.16 Update Element Action

The update element action API function call updates an element in Stage 1, thereby creating a new level for the element in Stage 1. Elements are updated only if there are differences between the incoming source in the FROM location and the target Stage 1 source.

Assembler: ENHAEUPD

COBOL: ECHAEUPD

2.16.1 AEUPD_RQ Request Structure Fields

Immediately following the header is the data area of the AEUPD_RQ request structure. The request structure is where you set your selection criteria.

All request selection fields are explained in the following table. For default values, see the *SCL Reference Guide*.

Field	Length	Description
AEUPD_RQ_ELM	Character 10	Element name
AEUPD_RQ_ELM_THRU	Character 10	Through element name
AEUPD_RQ_DDN	Character 8	File or DD name where member(s) reside. Either DDN or DSN must be specified, but not both.
AEUPD_RQ_DSN	Character 44	Data set name where member(s) reside. Either DDN or DSN must be specified, but not both.
AEUPD_RQ_MBR	Character 10	PDS member name
AEUPD_RQ_ENV	Character 8	Environment name
AEUPD_RQ_SYSTEM	Character 8	System name
AEUPD_RQ_SUBSYS	Character 8	Subsystem name
AEUPD_RQ_TYPE	Character 8	Type name
AEUPD_RQ_CCID	Character 12	Change control id
AEUPD_RQ_COMM	Character 40	Comment
AEUPD_RQ_DEL	Character 1	Delete input source (Y/N)
AEUPD_RQ_OVESIGNO	Character 1	Override signout (Y/N)
AEUPD_RQ_BYP_ GENPRO	Character 1	Bypass generate processor (Y/N)

Field	Length	Description
AEUPD_RQ_PROGRO	Character 8	Processor group name
AEUPD_RQ_RESERVE	Character 3	** Reserved field **

2.17 Enterprise Package Function

AllFusion CM Enterprise Workbench is a web based administrator's tool. This facility provides a way of linking Endeavor and AllFusion Harvest CM packages together. It consists of a web based front-end that enables you to perform package actions against AllFusion Harvest CM and/or Endeavor.

This feature includes fields for panels and storage structures to identify and allow enterprise package selection. It also includes tools to enforce restrictions on package actions.

2.17.1 Enterprise Package Fields

You can use include/exclude selection criteria on the Package Foreground Options Menu for enterprise packages. This feature operates just like the existing status selection flags. You can use these three values:

- A - List all packages.
- E - Limit the list to enterprise packages.
- X - Exclude enterprise packages from the list.

The default value is A. You can change the default by modifying the ENTERPRISE_PKG field in the Configuration table, ENDICNFG. Shown below is a copy of the modified panel.

```

----- Package Foreground Options Menu -----
Option ==>

 1 DISPLAY      - Display Package Information
 2 CREATE/MODIFY - Create or Modify Package
 3 CAST         - Prepare Package for Review
 4 REVIEW       - Approve or Deny Package
 5 EXECUTE      - Submit or Execute Package
 6 SHIP         - Ship Packages
 7 BACKOUT      - Perform Backout or Backin Processing
 8 COMMIT       - Clear Backout Information
 9 UTILITIES    - Reset, Delete, or Export Package
 L DistribuLink - Perform Product Collection Request

      Package ID ==>

Limit selection list options. These options are used by the
DISPLAY and UTILITIES functions:

      In-Edit..... Y           In-Execution... Y
      In-Approval.... Y       Executed..... Y
      Denied..... Y          Committed..... Y
      Approved..... Y         Enterprise Pkg.. A

```

A new field was added to the Display Package panel to indicate if the current package is an enterprise package. Possible values displayed in this field are:

- Y - This package is an enterprise package.

- N - This package is not an enterprise package.

Shown below is a copy of the modified panel.

```

----- PACKAGE DISPLAY -----
OPTION ==>

      blank - Display Action Summary      B - Display Backout Information
      A - Display Approvers              S - Display SCL
      R - Display Cast Report
      N - Display Package Notes
PACKAGE ID:  FHBDEC21                    STATUS:  IN-EDIT
DESCRIPTION:  tst
PACKAGE TYPE: STANDARD                   ENTERPRISE PACKAGE:  Y
SHARABLE PACKAGE:  N
BACKOUT ENABLED:  Y
EXECUTION WINDOW FROM:  22DEC00 00:00    TO:  31DEC79 00:00

      USER ID  DATE    TIME
CREATED:      BUCFR02  22DEC99  15:15
LAST UPDATED:
CAST:
APPROVED/DENIED:
EXECUTED:
BACKED OUT:
BACKED IN:
COMMITTED:
ENDEVOR RC:

```

Several enterprise package related fields exist in the ENHALPKG list package structure. The field ALPKG_RQ_ENTPFLG exists in the request structure data area. You can use this field to limit which packages are selected and returned to the calling program. The allowed values are:

- blank - If left blank, a default value of A is assumed.
- A - Select all packages.
- E - Limit the list to only enterprise packages.
- X - Exclude enterprise packages.

The field ALPKG_RS_FLG_ECOR exists in the ENHALPKG response structure data area. This field indicates if a package is an enterprise package. Possible values displayed in this field are:

- Y - This package is an enterprise package.
- N - This package is not an enterprise package.

A new field, PECBENPK, was added to the \$PECBDS Package Exit control structure to indicate if the current package is an enterprise package. Possible values displayed in this field are:

- Y - This package is an enterprise package.
- N - This package is not an enterprise package.

2.17.2 Enterprise Package Restrictions

API enterprise package rules exist to ensure the enterprise packages remain in sync. Listed below are the enterprise package restrictions:

- You can create or modify an Endeavor enterprise package only by using the Endeavor package facility.
- You can correlate a package to an AllFusion CM Enterprise Workbench package prior to execution by using AllFusion CM Enterprise Workbench. In other words, the package must be in a status prior to IN-EXEC in order to be correlated to a AllFusion CM Enterprise Workbench package.
- An Enterprise package can be comprised of multiple AllFusion Harvest CM packages and/or multiple Endeavor packages. A single AllFusion Harvest CM and/or single Endeavor package, however, can only be associated with one Enterprise package. For example, AllFusion Harvest CM packages HARPKG1 and HARPKG2 and Endeavor packages ENDPKG1 and ENDPKG2 can be associated with Enterprise package ENTPKG1, but HARPKG1 and ENDPKG1 can not also be associated with ENTPKG2.
- Cast a package using AllFusion CM Enterprise Workbench and/or the Endeavor package facility.
- Approve or deny a package using AllFusion CM Enterprise Workbench and/or the Endeavor package facility.
- You must use AllFusion CM Enterprise Workbench to execute, reset or delete a package.
- You cannot reset an Endeavor package if it is associated with an Enterprise package. You must first remove the correlation.
- Use the RESTRICT_BACKOUT option in the Optional Features Table to control the rules related to the backout and backin actions. By default, you can perform the backout and backin actions using AllFusion CM Enterprise Workbench or the Endeavor package facility. If you activate the RESTRICT_BACKOUT option, you can only use AllFusion CM Enterprise Workbench to backout or backin enterprise packages.
- Commit packages using AllFusion CM Enterprise Workbench or the Endeavor package facility.
- You can use AllFusion CM Enterprise Workbench to perform actions against Endeavor packages that are not correlated to a AllFusion CM Enterprise Workbench package.

If you request one of the restricted actions against an enterprise package and Endeavor determines AllFusion CM Enterprise Workbench did not initiate the request, error message PKMR410E is generated and the return and reason codes are set to 12 and 14 respectively. This is the text associated with the message:

PKMR410E Enterprise packages require the use of the AllFusion CM Enterprise Workbench facility.

If you initiated the request from one of the ISPF package panels, an 'action failed' short message appears in the panel message area. If you enter the PF1 key at this point, the long message text described above appears.

2.18 Approve Package

The approve package API function call allows you to approve a package.

Assembler: ENHAPAPP

COBOL: ECHAPAPP

2.18.1 APAPP_RQ Request Structure Fields

Field	Length	Description
APAPP_RQ_PKGID	Character 16	Package id. Wildcarding is not supported. Security rules and user exits play a part in whether the user is allowed to approve a package.
APAPP_RQ_RESERVE	Character 3	** Reserved field **

2.19 Backin Package

The backin package API function call allows you to backin a package.

Assembler: ENHAPBKI

COBOL: ECHAPBKI

2.19.1 APBKI_RQ Request Structure Fields

Field	Length	Description
APBKI_RQ_PKGID	Character 16	Package id. Wildcarding is not supported. Security rules and user exits play a part in whether the user is allowed to backin a package.
APBKI_RQ_RESERVE	Character 3	** Reserved field **

2.20 Backout Package

The backout package API function call allows you to backout a package.

Assembler: ENHAPBKO

COBOL: ECHAPBKO

2.20.1 APBKO_RQ Request Structure Fields

Field	Length	Description
APBKO_RQ_PKGID	Character 16	Package id. Wildcarding is not supported. Security rules and user exits play a part in whether the user is allowed to backout a package.
APBKO_RQ_RESERVE	Character 3	** Reserved field **

2.21 Cast Package

The cast package API function call allows you to cast a package. All the fields except the package id field are optional. If left blank, the existing values specified when the package was created or modified are used.

Assembler: ENHAPCAS

COBOL: ECHAPCAS

2.21.1 APCAS_RQ Request Structure Fields

Field	Length	Description
APCAS_RQ_PKGID	Character 16	Package id. Wildcarding is not supported. Security rules and user exits play a part in whether the user is allowed to cast a package.
APCAS_RQ_VALCMP	Character 1	<p>Validate Components indicates whether Endeavor should validate package components when casting the package. Valid values are:</p> <ul style="list-style-type: none"> ▪ Blank, use existing value from package header ▪ Y - validate components and do not allow the cast if validation fails ▪ N - do not validate components ▪ W - validate components, but do not fail the cast if there are errors
APCAS_RQ_BOENABLED	Character 1	<p>Backout enabled indicates whether the backout/backin facility is available for this package. Valid values are:</p> <ul style="list-style-type: none"> ▪ Blank, use existing value from package header ▪ Y - backout/backin facility is enabled ▪ N - backout/backin facility is not enabled

Field	Length	Description
APCAS_RQ_EWF_DATE	Character 7	Execution window from date indicates the time frame within which you can execute a package. Valid values are blank, use existing date from the package header, or a valid date in the format of DDMMYY (30JUN00). If you specify the from date, then you must specify the from time. If you leave the from date blank, the from time is ignored.
APCAS_RQ_EWF_TIME	Character 5	Execution window from time indicates the time frame within which you can execute a package. Valid values are blank, use existing time from the package header, or a valid time in the format of HH:MM (23:59). If you specify the from date, you must specify the from time. If you leave the from date blank, the from time is ignored.
APCAS_RQ_EWT_DATE	Character 7	Execution window to date indicates the time frame within which you can execute a package. Valid values are blank, use existing date from the package header, or a valid date in the format of DDMMYY (30JUN00). If you specify the to date, you must specify the to time. If you leave the to date blank, the to time is ignored.
APCAS_RQ_EWT_TIME	Character 5	Execution window to time indicates the time frame within which you can execute a package. Valid values are blank, use existing time from the package header, or a valid time in the format of HH:MM (23:59). If you specify the to date, you must specify the to time. If you leave the to date, the to time is ignored.
APCAS_RQ_RESERVE	Character 3	** Reserved field **
	Character 3	Alignment characters

Note: All cast execution messages are written to the package cast report. You can view this on-line or by executing an API List Cast action.

2.22 Commit Package

The commit package API function call allows you to commit a package.

Assembler: ENHAPCOM

COBOL: ECHAPCOM

2.22.1 APCOM_RQ Request Structure Fields

Field	Length	Description
APCOM_RQ_PKGID	Character 16	Package id. Wildcarding is not supported. Security rules and user exits play a part in whether the user is allowed to commit a package.
APCOM_RQ_RESERVE	Character 3	** Reserved field **

2.23 Define Package

The define package API function call allows you to create or modify a package.

For the modify function, if a field is not specified, the existing data is retained. This is also true for the SCL; if none of the COPY_PKGID, IMPORT_DDND or IMPORT_DSN fields are specified, the existing SCL is retained.

Assembler: ENHAPDEF

COBOL: ECHAPDEF

2.23.1 APDEF_RQ Request Structure Fields

Immediately following the header is the data area of the APDEF_RQ request structure. The request structure is where you set your selection criteria.

Field	Length	Description
APDEF_RQ_PKGID	Character 16	Package id. This field is required.
APDEF_RQ_FUNC	Character 1	Function. Valid values are (C)reate or (M)odify. This field is required.
APDEF_RQ_DESC	Character 50	Package description. This field is required for the create function.
APDEF_RQ_TYPE	Character 1	Type of package. Valid values are (S)tandard or (E)mergency. This field will default to S for the create function.
APDEF_RQ_SHARE	Character 1	Sharable package (Y/N). This field defaults to N for the create function.
APDEF_RQ_APPEND	Character 1	Append SCL to existing package. This is only valid for the modify function. One of the COPY FROM SCL or IMPORT FROM SCL must be specified in conjunction with this field. This field will default to N.

Note: One and only one of the COPY_PKGID, IMPORT_DDND or IMPORT_DSN fields must be specified in conjunction with the APPEND option.

Field	Length	Description
APDEF_RQ_BOENABLED	Character 1	Backout enabled flag (Y/N). This field will default to Y for the create function.
APDEF_RQ_EWF_DATE	Character 7	<p>Execution window from date (DDMMYY). This field is optional.</p> <p>If not specified and the function is modify, the existing from date and time are retained. If not specified and the function is create, the values default to the date and time this function is executed.</p>
APDEF_RQ_EWF_TIME	Character 5	Execution window from time (HH:MM). Valid values are 00:00 through 23:59.
APDEF_RQ_EWT_DATE	Character 7	<p>Execution window through date (DDMMYY). This field is optional.</p> <p>If not specified and the function is modify, the existing through date and time are retained. If not specified and the function is create, the values default to 31DEC79 and 00:00.</p> <p>You are allowed to specify the through date and time without specifying the from date and time for the create function. In this case, the from date and time default is used. In order to modify the through date and time, you must specify the from date and time.</p>
APDEF_RQ_EWT_TIME	Character 5	Execution window through time (HH:MM). Valid values are 00:00 through 23:59.
APDEF_RQ_COPY_PKGID	Character 16	<p>Copy SCL from package id.</p> <p>Note: For the create function, you must specify one and only one of the COPY_PKGID, IMPORT_DDN or IMPORT_DSN fields.</p>

Field	Length	Description
APDEF_RQ_IMPORT_DDND	Character 8	Import SCL from DD name. Note: For the create function, you must specify one and only one of the COPY_PKGID, IMPORT_DDND or IMPORT_DSN fields.
APDEF_RQ_IMPORT_DSN	Character 44	Import SCL from dataset name. Note: For the create function, you must specify one and only one of the COPY_PKGID, IMPORT_DDND or IMPORT_DSN fields.
APDEF_RQ_IMPORT_MBR	Character 10	Import SCL from member name. Only valid when an PDS import dataset name is specified.
APDEF_RQ_RMTSCL	Character 1	Import SCL file is remote.

2.23.1.1 Note Fields

Field	Length	Description
APDEF_RQ_NOTES1	Character 60	Notes line 1
APDEF_RQ_NOTES2	Character 60	Notes line 2
APDEF_RQ_NOTES3	Character 60	Notes line 3
APDEF_RQ_NOTES4	Character 60	Notes line 4
APDEF_RQ_NOTES5	Character 60	Notes line 5
APDEF_RQ_NOTES6	Character 60	Notes line 6
APDEF_RQ_NOTES7	Character 60	Notes line 7
APDEF_RQ_NOTES8	Character 60	Notes line 8
APDEF_RQ_RESERVE	Character 2	** Reserved field **

The note fields are processed in order (NOTES1-NOTES8) until a blank field is found. For the modify function, each field processed, replaces the entire existing 60 character value. If more notes exist than are coded, the values of the additional existing lines are retained.

Note: All DEFINE package execution messages are written to the package execution report file. The DD name of this file is also C1MSG1. Make sure a DD statement is specified in your JCL for this report file.

2.24 Delete Package

The delete package API function call allows you to delete a package. The older than clause and the package status clause are not currently supported.

Assembler: ENHAPDEL

COBOL: ECHAPDEL

2.24.1 APDEL_RQ Request Structure Fields

Field	Length	Description
APDEL_RQ_PKGID	Character 16	Package id. Wildcarding is not supported. Security rules and user exits play a part in whether the user is allowed to delete a package.
APDEL_RQ_RESERVE	Character 3	** Reserved field **

2.25 Deny Package

The deny package API function call allows you to deny approval of a package. The notes clause is not currently supported.

Assembler: ENHAPDEN

COBOL: ECHAPDEN

2.25.1 APDEN_RQ Request Structure Fields

Field	Length	Description
APDEN_RQ_PKGID	Character 16	Package id. Wildcarding is not supported. Security rules and user exits play a part in whether the user is allowed to deny approval of a package.
APDEN_RQ_RESERVE	Character 3	** Reserved field **

2.26 Execute Package

The execute package API function call allows you to execute a package. The execution window clause and the package status clause are not currently supported.

All execute action messages are written to the package execution report file. The DD name of this file is determined by the value specified in the AACTL_CMDMSG_DDN field. If you do not specify a value, a default value of C1EXMSGs is used. Make sure you specify a DD statement in your JCL for this report file.

Assembler: ENHAPEXE

COBOL: ECHAPEXE

2.26.1 APEXE_RQ Request Structure Fields

Field	Length	Description
APEXE_RQ_PKGID	Character 16	Package id. Wildcarding is not supported. Security rules and user exits play a part in whether the user is allowed to execute a package.
APEXE_RQ_RESERVE	Character 3	** Reserved field **

2.27 List Package Action Summary

The list package action summary API function allows you to list element actions associated with a package.

Assembler: ENHALSUM

COBOL: ECHALSUM

2.27.1 ALSUM_RQ Request Structure Fields

Field	Length	Description
ALSUM_RQ_PKGID	Character 16	Package id. Wildcarding is not supported. Security rules and user exits play a part in whether the package data is returned or not.
ALSUM_RQ_RESERVE	Character 3	** Reserved field **

2.27.2 ALSUM_RS Response Structure Fields

Immediately following the header is the data area of the ALSUM_RS response structure. This response structure maps the action summary data associated with a package. The information contained in the response structure is explained in the following table:

Field	Length	Description
ALSUM_RS_PKGID	Character 16	Package id.
ALSUM_RS_STMTNO	Character 5	SCL statement number. Value is right justified and padded with spaces.
ALSUM_RS_SITE	Character 1	Site id.
ALSUM_RS_ACTN	Character 8	Element action.
ALSUM_RS_NDVRRRC	Character 4	Highest Endeavor return code encountered.
ALSUM_RS_PROCRRC	Character 4	Highest processor return code encountered.
ALSUM_RS_BEXD	Character 7	Beginning of execution date.
ALSUM_RS_BEXT	Character 5	Beginning of execution time.
ALSUM_RS_EEXD	Character 7	End of execution date.
ALSUM_RS_EEXT	Character 5	End of execution time.

Field	Length	Description
ALSUM_RS_CCID	Character 12	CCID associated with this action.
ALSUM_RS_COMM	Character 40	Comment associated with this action.
ALSUM_RS_RUD	Character 7	Last physical record update date.
ALSUM_RS_RUT	Character 5	Last physical record update time.
ALSUM_RS_RUU	Character 8	Userid associated with last physical record update.

2.27.2.1 Source Location Information

Field	Length	Description
ALSUM_RS_SAPREQ	Character 1	Approval required flag. Possible value are (Y)es or blank.
ALSUM_RS_SVAREQ	Character 1	Validation required flag. Possible value are (Y)es or blank.
ALSUM_RS_SLOC	Character 1	Source location. Possible values are: 'C' Endeavor location 'F' File or DDNAME 'D' Data set 'A' Endeavor archive 'P' Path

2.27.2.2 Data Available When Location is C or A

Field	Length	Description
ALSUM_RS_SSITE	Character 1	Site id
ALSUM_RS_SENV	Character 8	Environment
ALSUM_RS_SSYS	Character 8	System
ALSUM_RS_SSBS	Character 8	Subsystem
ALSUM_RS_SELMOFF	BINARY (2 bytes)	Element area offset from beginning of record.
ALSUM_RS_STYP	Character 8	Type
ALSUM_RS_SSTG	Character 1	Stage number (1 or 2)

Field	Length	Description
ALSUM_RS_SSTGN	Character 8	Stage name
ALSUM_RS_SSTGI	Character 1	Stage id
ALSUM_RS_SVLL	Character 5	Version/level number. Format is 'VV.LL'.
ALSUM_RS_SDD	Character 7	Last delta level date
ALSUM_RS_SDT	Character 5	Last delta level time
ALSUM_RS_SGD	Character 7	Last generate date
ALSUM_RS_SGT	Character 5	Last generate time
ALSUM_RS_SPD	Character 7	Last processor date
ALSUM_RS_SPT	Character 5	Last processor time
ALSUM_RS_SPPKGID	Character 16	Previous package id associated with source.
ALSUM_RS_SPPKGTS	BINARY (8 bytes)	Cast timestamp associated with source. Used to provide package id uniqueness.

2.27.2.3 Data Available When Location is D, F or P

Field	Length	Description
ALSUM_RS_SFILEOFF	BINARY (2 bytes)	Offset from beginning of this record to the dataset or path name area.
ALSUM_RS_SNAMEOFF	BINARY (2 bytes)	Offset from beginning of this record to the member or file name area.
ALSUM_RS_SPPKGID	Character 16	Previous package ID associated with source.
ALSUM_RS_SPPKGTS	BINARY (8 bytes)	Cast timestamp associated with source. Used to provide package ID uniqueness.

2.27.2.4 Target Location Information

Field	Length	Description
ALSUM_RS_TAPREQ	Character 1	Approval required flag. Possible value are (Y)es or blank.

Field	Length	Description
ALSUM_RS_TVAREQ	Character 1	Validation required flag. Possible value are (Y)es or blank.
ALSUM_RS_TLOC	Character 1	Target location. Possible values are: 'C' Endeavor location 'F' File or DDNAME 'D' Data set 'A' Endeavor archive

2.27.2.5 Data Available When Location is C or A

Field	Length	Description
ALSUM_RS_TSITE	Character 1	Site id
ALSUM_RS_TENV	Character 8	Environment
ALSUM_RS_TSYS	Character 8	System
ALSUM_RS_TSBS	Character 8	Subsystem
ALSUM_RS_TELMOFF	BINARY (2 bytes)	Element area offset from beginning of record
ALSUM_RS_TTYP	Character 8	Type
ALSUM_RS_TSTG	Character 1	Stage number (1 or 2)
ALSUM_RS_TSTGN	Character 8	Stage name
ALSUM_RS_TSTGI	Character 1	Stage id
ALSUM_RS_TVLL	Character 5	Version/level number. Format is 'VV.LL'.
ALSUM_RS_TDD	Character 7	Last delta level date
ALSUM_RS_TDT	Character 5	Last delta level time
ALSUM_RS_TGD	Character 7	Last generate date
ALSUM_RS_TGT	Character 5	Last generate time
ALSUM_RS_TPD	Character 7	Last processor date
ALSUM_RS_TPT	Character 5	Last processor time
ALSUM_RS_TPPKGID	Character 16	Previous package id associated with target.

Field	Length	Description
ALSUM_RS_TPPKGTS	BINARY (8 bytes)	Cast timestamp associated with target. Used to provide package id uniqueness.

2.27.2.6 Data Available When Location is D, F or P

Field	Length	Description
ALSUM_RS_TFILEOFF	BINARY (2 bytes)	Offset from beginning of this record to the dataset or path name area.
ALSUM_RS_TNAMEOFF	BINARY (2 bytes)	Offset from beginning of this record to the member or file name area.
ALSUM_RS_TPPKGID	Character 16	Previous package id associated with target.
ALSUM_RS_TPPKGTS	BINARY (8 bytes)	Cast timestamp associated with target. Used to provide package id uniqueness.

2.27.2.7 Data Available When Location is: C, A, D, F or P

Field	Length	Description
ALSUM_RS_DAREAS	BINARY (2600 bytes)	Data area buffer space for element, file and name.

2.28 List Package Approvers

The list package approvers API function allows you to produce a list of approvers for a package.

Assembler: ENHALAPP

COBOL: ECHALAPP

2.28.1 ALAPP_RQ Request Structure Fields

Immediately following the header is the data area of the ALAPP_RQ request structure. The request structure is where you set your selection criteria. All request selection fields are explained in the following table.

Field	Length	Description
ALAPP_RQ_PKGID	Character 16	Package id. Wildcarding is not supported. Security rules and user exits play a part in whether the package data is returned or not.
ALAPP_RQ_RESERVE	Character 3	** Reserved field **

2.28.2 ALAPP_RS Response Structure Fields

Immediately following the header is the data area of the ALAPP_RS response structure. This response structure maps the approver data associated with a package. The information contained in the response structure is explained in the following table:

Field	Length	Description
ALAPP_RS_PKGID	Character 16	Package id
ALAPP_RS_GRPNM	Character 16	Approver group name.
ALAPP_RS_ENVNM	Character 8	Approver group's environment.
ALAPP_RS_OUNIQ	BINARY (8 bytes)	Approver group overflow uniqueness.
ALAPP_RS_SEQNO	BINARY (4 bytes)	Record sequence number.
ALAPP_RS_SITE	Character 1	Site id.

Field	Length	Description
ALAPP_RS_APFLG	Character 1	Overall approval status. Possible values are: <ul style="list-style-type: none"> ▪ Blank - approval pending ▪ A - approved ▪ D - denied
ALAPP_RS_GRPTYP	Character 1	Approval group type. Possible values are (S)tandard or (E)xternal. External groups are groups defined using an external security product such as CA-ACF2 or CA-TopSecret.
	Character 1	Alignment character.
ALAPP_RS_QUORUM	BINARY (2 bytes)	Quorum count. Minimum number of approvers that must approve this package.
ALAPP_RS_NOAPPR	BINARY (2 bytes)	Number of approvers in this group. For external groups, this number starts at zero and is incremented each time an approver approves or denies the package. A maximum of 16 approvers may exist per record.
ALAPP_RS_RUD	Character 7	Last physical record update date.
ALAPP_RS_RUT	Character 5	Last physical record update time.
ALAPP_RS_RUU	Character 8	Userid associated with last physical record update.
ALAPP_RS_APID	Character 8	Userid of approver.
ALAPP_RS_APD	Character 7	Approval/denial date.
ALAPP_RS_APT	Character 5	Approval/denial time.
ALAPP_RS_APFLAG	Character 8	Approval status. Possible values are blank (approval pending), APPROVED or DENIED.
ALAPP_RS_APREQD	Character 8	Approval required indicator. Possible values are REQUIRED--approval or this user is required, or blank--approval of this user is optional.

2.29 List Package Backout Information

The list package backout information API function allows you to list the backout information associated with a package.

Assembler: ENHALBKO

COBOL: ECHALBKO

2.29.1 ALBKO_RQ Request Structure Fields

Field	Length	Description
ALBKO_RQ_PKGID	Character 16	Package id. Wildcarding is not supported. Security rules and user exits play a part in whether the package data is returned or not.
ALBKO_RQ_RESERVE	Character 3	** Reserved field **

2.29.2 ALBKO_RS Response Structure Fields

Immediately following the header is the data area of the ALBKO_RS response structure. This response structure maps the backout data associated with a package. The information contained in the response structure is explained in the following table:

Field	Length	Description
ALBKO_RS_PKGID	Character 16	Package id.
ALBKO_RS_SITE	Character 1	Site id.
ALBKO_RS_RUD	Character 7	Last physical record update date.
ALBKO_RS_RUT	Character 5	Last physical record update time.
ALBKO_RS_RUU	Character 8	Userid associated with last physical record update.
ALBKO_RS_LOC	Character 1	Source/target location flag. Possible values are: 'S' record is associated with source location. 'T' record is associated with target location.

Field	Length	Description
ALBKO_RS_DIR	Character 1	Backout direction. Possible values are: 'I' member has been backed-in. 'O' member has been backed-out.
ALBKO_RS_DIRLIT	Character 10	Backout direction literal. Possible values are: 'BACKED-IN' member has been backed-in. 'BACKED-OUT' member has been backed-out.
ALBKO_RS_MBR	Character 8	Current member name.
ALBKO_RS_MBRNEW	Character 8	New member name.
ALBKO_RS_MBRSAVE	Character 8	Saved member name in binary format.
ALBKO_RS_DSN	Character 44	Backout data set name.
	Character 3	Alignment characters.

2.30 List Package Cast Report

The list package cast report API function allows you to list a cast report associated with a package.

Assembler: ENHALCAS

COBOL: ECHALCAS

2.30.1 ALCAS_RQ Request Structure Fields

Field	Length	Description
ALCAS_RQ_PKGID	Character 16	Package id. Wildcarding is not supported. Security rules and user exits play a part in whether the package data is returned or not.
ALCAS_RQ_RESERVE	Character 3	** Reserved field **

2.30.2 ALCAS_RS Response Structure Fields

Immediately following the header is the data area of the ALCAS_RS response structure. This response structure maps the cast report data associated with a package. The information contained in the response structure is explained in the following table:

Field	Length	Description
ALCAS_RS_PKGID	Character 16	Package id.
ALCAS_RS_SITE	Character 1	Site id.
ALCAS_RS_RUD	Character 7	Last physical record update date.
ALCAS_RS_RUT	Character 5	Last physical record update time.
ALCAS_RS_RUU	Character 8	Userid associated with last physical record update.
	Character 3	Alignment characters.
ALCAS_RS_SEQNO	BINARY (4 bytes)	Report line sequence number, starting with zeros and incremented by 1.
ALCAS_RS_RPT	Character 133	Cast report line.
	Character 3	Alignment characters.

2.31 List Package Correlation

The list package correlation API function allows you to list correlation records associated with a package.

Assembler: ENHALCOR

COBOL: ECHALCOR

2.31.1 ALCOR_RQ Request Structure Fields

Field	Length	Description
ALCOR_RQ_PKGID	Character 16	Package id. Wildcarding is not supported. Security rules and user exits play a part in whether the package data is returned or not.
ALCOR_RQ_TYPE	Character 1	Correlation type. Possible values are: <ul style="list-style-type: none"> ■ H - Enterprise Package ■ I - Infoman ■ U - User defined
ALCOR_RQ_RESERVE	Character 3	** Reserved field **

2.31.2 ALCOR_RS Response Structure Fields

Immediately following the header is the data area of the ALCOR_RS response structure. This response structure maps correlation data associated with a package. The information contained in the response structure is explained in the following table:

Field	Length	Description
ALCOR_RS_PKGID	Character 16	Package id.
ALCOR_RS_TYPE	Character 1	Correlation type. Possible values are: <ul style="list-style-type: none"> H - Enterprise package I - Infoman U - user defined
	Character 3	Alignment characters.
ALCOR_RS_SEQNO	BINARY (4 bytes)	Record sequence number.

2.31 List Package Correlation

Field	Length	Description
ALCOR_RS_SITE	Character 1	Site id.
ALCOR_RS_RUD	Character 7	Last physical record update date.
ALCOR_RS_RUT	Character 5	Last physical record update time.
ALCOR_RS_RUU	Character 8	Userid associated with last physical record update.
ALCOR_RS_CORRID	Character 32	Correlation id.
ALCOR_RS_DATA	Character 80	Correlation data.
	Character 3	Alignment characters.

2.32 List Package Header

The list package header API function allows you to produce a list of packages and the data associated with each.

Assembler: ENHALPKG

COBOL: ECHALPKG

2.32.1 ALPKG_RQ Request Structure Fields

Field	Length	Description
ALPKG_RQ_PKGID	Character 16	Package id. Wildcarding is supported. Security rules and user exits play a part in which records are returned.
ALPKG_RQ_PKG_TYPE	Character 1	Package type selection criteria. Valid values are blank, (S)tandard or (E)mergency. If left blank, both types of packages are eligible for selection.
ALPKG_RQ_IN_EDIT	Character 1	Package status selection criteria. Valid values are blank, Y or N. If you want to eliminate packages in this status, specify a value of N in this field. If left blank, a default value of Y is assumed.
ALPKG_RQ_IN_APPR	Character 1	Package status selection criteria. Valid values are blank, Y or N. If you want to eliminate packages in this status, specify a value of N in this field. If left blank, a default value of Y is assumed.
ALPKG_RQ_DENIED	Character 1	Package status selection criteria. Valid values are blank, Y or N. If you want to eliminate packages in this status, specify a value of N in this field. If left blank, a default value of Y is assumed.

Field	Length	Description
ALPKG_RQ_APPROVED	Character 1	Package status selection criteria. Valid values are blank, Y or N. Blank and Y produce the same results. If you want to eliminate packages in this status, specify a value of N in this field. If left blank, a default value of Y is assumed.
ALPKG_RQ_IN_EXEC	Character 1	Package status selection criteria. Valid values are blank, Y or N. If you want to eliminate packages in this status, specify a value of N in this field. If left blank, a default value of Y is assumed.
ALPKG_RQ_EXEC	Character 1	Package status selection criteria. Valid values are blank, Y or N. If you want to eliminate packages in this status, specify a value of N in this field. If left blank, a default value of Y is assumed.
ALPKG_RQ_COMMIT	Character 1	Package status selection criteria. Valid values are blank, Y or N. If you want to eliminate packages in this status, specify a value of N in this field. If left blank, a default value of Y is assumed.
ALPKG_RQ_ENTPFLG	Character 1	Enterprise package flag. Valid values are blank, (A)ll, (E)nterprise only or e(X)clude enterprise packages. If left blank, a default value of A is assumed.

Field	Length	Description
ALPKG_RQ_DTE_TYPE	Character 2	Package 'older than days' date type selection criteria. Valid values are blank, (CR)eate, (MO)dify, (CA)st, (AP)rove/deny, (EX)ecute, (BO)backout, (BI)backin or (CO)mmit. This field is used in conjunction with the days old field. This date type in the package record is compared against the current date to determine if the date is older than the number of days specified. If this field is blank and/or the days old field is zero or blank, selection is not affected.
ALPKG_RQ_DAYS_OLD	Character 3	Package 'older than days' selection criteria. Valid values are blank and 0-999. The API right justifies and zero fills this field as necessary. This field is used in conjunction with the date type field. The date type in the package record is compared against the current date to determine if the date is older than this number of days. If this field is blank and/or the date type field is zero or blank, selection is not affected.

Field	Length	Description
ALPKG_RQ_APPRID	Character 8	<p>Package approver selection criteria. Valid values are blank or a fully qualified user id. This selection criteria is applied after all other criteria; package id, package type, package status, and days old. If this approver id appears in any of the groups associated with a package, the package is selected regardless of whether the user has already approved/denied the package or not.</p> <p>The API may be unable to apply the selection approver id against all the approver groups associated with a package. This condition only occurs under the following conditions:</p> <ul style="list-style-type: none"> ■ The selection approver id is specified and it is different than the TSO/jobname user id and the approver group is an external approver group. ■ The API issues message API0091W each time this condition is encountered. ■ The message identifies the package id and approver group name. ■ The selection process continues with the next approver group.
ALPKG_RQ_FLAG	Character 1	Data request flag. It determines the amount of package header information written to the response block. Valid values are (blank) return all fields and (B) return only basic fields. If you enable this option, use the ALPKB_RS structure in ENHALPKG to map the response fields.
ALPKG_RQ_RESERVE	Character 2	** Reserved field **
	Character 3	Alignment characters

2.32.2 List Package Response Structure Fields

There are two types of response structures. The structure type depends on the value you specified in the ALPKG_RQ_FLAG field. If you specified 'B', use the ALPKB structure. Regarding your request type for the response structure, always specify the base response structure, ALPKG_RS, as the last parameter in the call to the API (ENASNDVR).

2.32.3 ALPKG_RS Response Structure Fields

Immediately following the header is the data area of the ALPKG_RS response structure. The information contained in the response structure is explained in the following table:

Field	Length	Description
ALPKG_RS_PKGID	Character 16	Package id.
ALPKG_RS_SITE	Character 1	Site id.
ALPKG_RS_COMMENT	Character 50	Comment or description associated with this package.
ALPKG_RS_PKG_TYPE	Character 10	Package type selection criteria. Possible values are STANDARD or EMERGENCY.
ALPKG_RS_PSHR	Character 1	Package shr option. Possible values are: 'Y' anyone can edit or cast package. 'N' only the creator can edit or cast package.
ALPKG_RS_BOFLG	Character 1	Package backout enabled flag. Possible values are: 'Y' backout enabled. 'N' backout is not enabled.
ALPKG_RS_WSD	Character 7	Execution window start date.
ALPKG_RS_WST	Character 5	Execution window start time.
ALPKG_RS_WED	Character 7	Execution window end date.
ALPKG_RS_WET	Character 5	Execution window end time.
ALPKG_RS_STAT	Character 12	Package status. Possible values are IN-EDIT, IN-APPROVAL, DENIED, APPROVED, IN-EXECUTION, EXECUTED, EXEC-FAILED or COMMITTED.

Field	Length	Description
ALPKG_RS_FLG_AGRE	Character 1	Approver group records exist flag. Possible values are (Y)es or (N)o.
ALPKG_RS_FLG_CARE	Character 1	Cast report records exist flag. Possible values are (Y)es or (N)o.
ALPKG_RS_FLG_CORE	Character 1	Correlation records exist flag. Possible values are (Y)es or (N)o.
ALPKG_RS_FLG_ECOR	Character 1	Enterprise correlation record flag. Indicates whether or not this package as an enterprise package. Possible values are (Y)es or (N)o.
ALPKG_RS_FLG_BONE	Character 1	Backout not in effect flag. Possible values are (Y)es or (N)o.
ALPKG_RS_FLG_BORE	Character 1	Backout records exist flag. Possible values are (Y)es or (N)o.
ALPKG_RS_FLG_ABN	Character 1	Execution of package abnormally terminated flag. Possible values are (Y)es or (N)o.
ALPKG_RS_FLG_PBO	Character 1	Package has been backed-out flag. Possible values are (Y)es or (N)o.
ALPKG_RS_FLG_BBHB	Character 1	Package backin/backout has begun flag. Possible values are (Y)es or (N)o.
ALPKG_RS_FLG_SAF	Character 1	External SAF flag. Possible values are (Y)es or (N)o.
ALPKG_RS_EXRC	Character 4	Execution return code. Possible values are blank, 0000, 0004, 0012 or 0016. This field is blank until the package is executed.
ALPKG_RS_BSTAT	Character 10	Package backout status. Possible values are BACKEDOUT or blanks.
	Character 1	Alignment character.
ALPKG_RS_NOCORR	BINARY (2 bytes)	Number of correlations defined.
ALPKG_RS_CRD	Character 7	Date package was created.
ALPKG_RS_CRT	Character 5	Time package was created.
ALPKG_RS_CRU	Character 8	Userid associated with the create action.

Field	Length	Description
ALPKG_RS_MOD	Character 7	Date package SCL was last modified.
ALPKG_RS_MOT	Character 5	Time package SCL was last modified.
ALPKG_RS_MOU	Character 8	Userid associated with the SCL update action.
ALPKG_RS_CAD	Character 7	Date package was cast.
ALPKG_RS_CAT	Character 5	Time package was cast.
ALPKG_RS_CAU	Character 8	Userid associated with the cast action.
ALPKG_RS_APD	Character 7	Date of final approval/denied of package.
ALPKG_RS_APT	Character 5	Time of final approval/denied of package.
ALPKG_RS_EXBD	Character 7	Package execution begin date.
ALPKG_RS_EXBT	Character 5	Package execution begin time.
ALPKG_RS_EXED	Character 7	Package execution end date.
ALPKG_RS_EXET	Character 5	Package execution end time.
ALPKG_RS_EXU	Character 8	Userid associated with the execute action.
ALPKG_RS_BOD	Character 7	Date package was backed-out.
ALPKG_RS_BOT	Character 5	Time package was backed-out.
ALPKG_RS_BOU	Character 8	Userid associated with the backout action.
ALPKG_RS_BID	Character 7	Date package was backed-in.
ALPKG_RS_BIT	Character 5	Time package was backed-in.
ALPKG_RS_BIU	Character 8	Userid associated with the backin action.
ALPKG_RS_COD	Character 7	Date package was committed.
ALPKG_RS_COT	Character 5	Time package was committed.
ALPKG_RS_COU	Character 8	Userid associated with the committed action.
ALPKG_RS_RUD	Character 7	Date package header record was last updated.

Field	Length	Description
ALPKG_RS_RUT	Character 5	Time package header record was last updated.
ALPKG_RS_RUU	Character 8	Userid associated with the last update.
ALPKG_RS_NOTE1	Character 60	Package note record.
ALPKG_RS_NOTE2	Character 60	Package note record.
ALPKG_RS_NOTE3	Character 60	Package note record.
ALPKG_RS_NOTE4	Character 60	Package note record.
ALPKG_RS_NOTE5	Character 60	Package note record.
ALPKG_RS_NOTE6	Character 60	Package note record.
ALPKG_RS_NOTE7	Character 60	Package note record.
ALPKG_RS_NOTE8	Character 60	Package note record.
	Character 2	Alignment characters.

2.32.4 ALPKB_RS Response Structure Fields

There are two types of response structures. The structure type depends on the value you specified in the ALPKG_RQ_FLAG field. If you specified 'B', use the ALPKB structure. Regarding your request type for the the response structure, always specify the base response structure, ALPKG_RS, as the last parameter in the call to the API (ENASNDVR).

Immediately following the header is the data area of the ALPKG_RS response structure. The information contained in the response structure is explained in the following table:

Field	Length	Description
ALPKB_RS_PKGID	Character 16	Package id
ALPKB_RS_COMMENT	Character 50	Comment or description
ALPKB_RS_PKG_TYPE	Character 10	Package types: <ul style="list-style-type: none"> ▪ Standard ▪ Emergency

Field	Length	Description
ALPKB_RS_STAT	Character 12	Package statuses: <ul style="list-style-type: none">▪ In-edit▪ In-approval▪ Denied▪ Approved▪ In-execution▪ Executed▪ Exec-failed▪ Committed

2.33 List Package SCL

The list package SCL API function allows you to list the SCL associated with a package.

Assembler: ENHALSCL

COBOL: ECHALSCL

2.33.1 ALSCL_RQ Request Structure Fields

Field	Length	Description
ALSCL_RQ_PKGID	Character 16	Package id. Wildcarding is not supported. Security rules and user exits play a part in whether the package data is returned or not.
ALSCL_RQ_RESERVE	Character 3	** Reserved field **
	Character 1	Alignment character

2.33.2 ALSCL_RS Response Structure Fields

Immediately following the header is the data area of the ALSCL_RS response structure. This response structure maps the SCL data associated with a package. The information contained in the response structure is explained in the following table:

Field	Length	Description
ALSCL_RS_PKGID	Character 16	Package id.
ALSCL_RS_SITE	Character 1	Site id.
ALSCL_RS_RUD	Character 7	Last physical record update date.
ALSCL_RS_RUT	Character 5	Last physical record update time.
ALSCL_RS_RUU	Character 8	Userid associated with last physical record update.
	Character 1	Alignment character.
ALSCL_RS_NOSTMTS	BINARY (2 bytes)	Number of SCL statements in this record. A maximum of 10 statements may exist in one record.
	Character 2	Alignment characters.

Field	Length	Description
ALSCL_RS_SCL	Character 80	Package SCL. This is a fixed length area of 800 characters. If the number of SCL statement and/or clauses (NOSTMTS) is less than 10, the remaining 80 character blocks are initialized to spaces. It is possible to have more than one statement with a single record.
	Character 2	Alignment characters.

2.34 Package Correlation

The package correlation API function allows you to create, delete or modify a correlation record associated with a package.

Assembler: ENHAPCOR

COBOL: ECHAPCOR

2.34.1 APCOR_RQ Request Structure Fields

Field	Length	Description
APCOR_RQ_PKGID	Character 16	Package id. Wildcarding is not supported. Security rules and user exits are not enforced for this action since it is an internal function.
APCOR_RQ_FUNC	Character 1	Function requested. Possible values are: 'C' create a correlation record 'D' delete a correction record 'M' modify the DATA field value of an existing correlation record
APCOR_RQ_TYPE	Character 1	Correlation type. Possible values are: 'H' Enterprise package 'T' Infoman 'U' User defined
APCOR_RQ_CORRID	Character 32	Correlation id. Id of the Enterprise package, Infoman or User defined entity associated with this package.
APCOR_RQ_DATA	Character 80	Free form data, depending of the application.
APCOR_RQ_RESERVE	Character 3	** Reserved field **
	Character 3	Alignment characters

2.35 Reset Package

The reset package API function allows you to reset a package.

Assembler: ENHAPRES

COBOL: ECHAPRES

2.35.1 APRES_RQ Request Structure Fields

Field	Length	Description
APRES_RQ_PKGID	Character 16	Package id. Wildcarding is not supported. Security rules and user exits play a part in whether the user is allowed to reset a package.
APRES_RQ_RESERVE	Character 3	** Reserved field **

2.36 Submit Package Request

The submit package request API function call defines the parameters necessary to submit a batch package job to the internal reader or to CA-7. Specify the jobcard location information, along with the 'TO' information and the appropriate action options. The API builds the appropriate jobstream and submits the job for execution.

Assembler: ENHAPSUB

COBOL: ECHAPSUB

2.36.1 APSUB_RQ Request Structure Fields

Immediately following the header is the data area of the APSUB_RQ request structure. The request structure is where you set your selection criteria.

Field	Length	Description
APSUB_RQ_PKGID	CHAR 16	Package id of the package you wish to submit. Wildcarding is permitted.

2.36.1.1 Jobcard Location Information

Field	Length	Description
APSUB_RQ_JCDDN	CHAR 8	File or DD name where a valid jobcard can be found. Either DDN or DSN must be specified, but not both.
APSUB_RQ_JCDSN	CHAR 44	Data set name where a valid jobcard can be found. Either DDN or DSN must be specified, but not both.
APSUB_RQ_JCMBR	CHAR 8	Member name where the jobcard can be found. This field is used in conjunction with the DSN field.
	CHAR 2	** Reserved field **

2.36.1.2 Submit TO Location Information

Field	Length	Description
APSUB_RQ_TOLOC	CHAR 8	Submit to location. Either enter an internal reader DDNAME or the literal CA7.

2.36.1.3 Action Options

Field	Length	Description
APSUB_RQ_WSTATUS	CHAR 1	Where current package status. Valid values are: A or blank - Approved F - Execute Failed B - Both (approved or execute failed)
APSUB_RQ_MULTJS	CHAR 1	Multiple jobstreams. Submit a unique job for each package. This parameter comes into play if you wildcard the pack age id field. Valid values are: Y - Yes N or blank - No. Note: This field is ignored for CA7 processing.
APSUB_RQ_INCRJN	CHAR 1	Increment jobname. Increment the last character in the jobcard you provide. Use this option in conjunction with the multiple jobstream field to increment the last character in the JCL jobcard for each job stream you submit. Valid values are: Y or blank - Yes N - No Note: This field is ignored for CA7 processing.

Field	Length	Description
APSUB_RQ_JCLPROC	CHAR 8	JCL procedure name. Identifies the name of the JCL procedure you wish to invoke in the submit package action JCL. If you do not specify a procedure name, the Endeavor procedure is invoked.

2.36.1.4 CA7 Action Options

Field	Length	Description
APSUB_RQ_DEPJN	CHAR 8	Dependent job name. To make this job dependent on another CA7 job, enter the job name of the dependent job name.
	CHAR 7	** Reserved field **

2.37 List Approver Group

The list approver group API function call extracts information about the approver group from the MCF that satisfies the criteria you define in the ALAGR_RQ request structure. Depending on the response file DDN (AACTL_LIST_DDND) field value, the first of the last response structure is placed in your defined response area, ALAGR_RS. Refer to 2.2, “Control Structure” on page 2-3 for details. This allows you to check the response quickly if you are looking for a specific approver group. The API also writes all responses generated by your request to an external data set if you specified a file output DD name in the control structure.

Assembler: ENHALAGR

COBOL: ECHALAGR

2.37.1 ALAGR_RQ Request Structure Fields

Immediately following the header is the data area of the ALAGR_RQ request structure. The request structure is where you set your selection criteria. All request selection fields are explained in the following table.

Field	Length	Description
ALAGR_RQ_RETURN	Character 1	F - for return only the first record that satisfies the request. A - for return all records that satisfy the request.
ALAGR_RQ_ENV	Character 8	Environment name. You cannot specify a wildcard character in this field.
ALAGR_RQ_AGRNAME	Character 16	Approver group name. This field can contain a wildcard character.

Searching: Map searching is not available for this request. The location value, Environment, must be explicitly specified. The approver group name can contain a wildcard.

2.37.2 ALAGR_RS Response Structure Fields

Immediately following its header is the data area of the ALAGR_RS response structure. The information contained in the response structure is explained in the following table.

2.37 List Approver Group

Field	Length	Description
ALAGR_RS_SITE	Character 1	Site id
ALAGR_RS_ENV	Character 8	Environment name
ALAGR_RS_STG_NAME	Character 8	Stage name
ALAGR_RS_STG_ID	Character 1	Stage id
ALAGR_RS_STG_NUM	Character 1	Stage number
ALAGR_RS_AGRNAME	Character 16	Approver group name
ALAGR_RS_UPD_CNT	Zoned Char 8	Record update count
ALAGR_RS_UPD_DATE	Zoned Char 8	Update date YYYYMMDD
ALAGR_RS_UPD_TIME	Zoned Char 8	Update time HHMMSSSTT
ALAGR_RS_UPD_USER	Character 8	Update user id
ALAGR_RS_FMID	Zoned Char 5	Record created release id
ALAGR_RS_QUORM	Zoned Char 8	Quorum count
ALAGR_RS_TITLE	Character 50	Title
ALAGR_RS_AUSER	16 * Character 8	16 Approver user id's
ALAGR_RS_AUREQ	16 * Character 1	16 Approver Required flags - blank or 'Y'

2.38 List Approver Group Junctions

The list approver group junctions API function call extracts information about the approver group from the MCF that satisfies the criteria you define in the ALAGJ_RQ request structure. Depending on the response file DDN (AACTL_LIST_DDN) field value, the first of the last response structure is placed in your defined response area, ALAGJ_RS. Refer to 2.2, “Control Structure” on page 2-3 for details. This allows you to check the response quickly if you are looking for a specific approver group junction. The API also writes all responses generated by your request to an external data set if you specified a file output DD name in the control structure.

Assembler: ENHALAGJ

COBOL: ECHALAGJ

2.38.1 ALAGJ_RQ Request Structure Fields

Immediately following the header is the data area of the ALAGJ_RQ request structure. The request structure is where you set your selection criteria. All request selection fields are explained in the following table.

Field	Length	Description
ALAGJ_RQ_RETURN	Character 1	F - for return only the first record that satisfies the request. A - for return all records that satisfy the request.
ALAGJ_RQ_ENV	Character 8	Environment name. This field cannot contain a wildcard character.
ALAGJ_RQ_SYSTEM	Character 8	System name. This field may be wildcarded by leaving it blank. If the system name is specified, it must be explicit.
ALAGJ_RQ_SUBSYS	Character 8	Subsystem name. This field may be wildcarded by leaving it blank. If the subsystem name is specified, it must be explicit.
ALAGJ_RQ_TYPE	Character 8	Type name. This field may be wildcarded by leaving it blank. If the type name is specified, it must be explicit.

Field	Length	Description
ALAGJ_RQ_STG_NUM	Character 1	Stage number. This field may be wildcarded by leaving it blank. If the stage number name is specified, it must be explicit.

Searching: Map searching is not available for this request because these key values can be stored in the MCF with the wild card values. The location value, Environment, must be explicitly specified. The other keys (System, Subsystem, Type, and Stage) may be individually wildcarded by leaving the field blank. Otherwise, the values specified for these fields are treated as explicit key values.

2.38.2 ALAGJ_RS Response Structure Fields

Immediately following its header is the data area of the ALAGJ_RS response structure. The information contained in the response structure is explained in the following table.

Field	Length	Description
ALAGJ_RS_SITE	Character 1	Site id
ALAGJ_RS_ENV	Character 8	Environment name
ALAGJ_RS_SYSTEM	Character 8	System name
ALAGJ_RS_SUBSYS	Character 8	Subsystem name
ALAGJ_RS_TYPE	Character 8	Type name
ALAGJ_RS_STG_NUM	Character 1	Stage number
ALAGJ_RS_UPD_CNT	Zoned Char 8	Record update count
ALAGJ_RS_UPD_DATE	Zoned Char 8	Update date YYYYMMDD
ALAGJ_RS_UPD_TIME	Zoned Char 8	Update time HHMMSSSTT
ALAGJ_RS_UPD_USER	Character 8	Update User id
ALAGJ_RS_FMID	Zoned Char 5	Record created release id
ALAGJ_RS_JUN_TYPE	Character 2	Junction type: 'ST' - Standard 'EM' - Emergency
ALAGJ_RS_AGNME	Character 16	Approver Group name

2.39 List Components/Where-used

The list components API function call allows you to produce a component list for an element or a 'where used' list for an element, member, related object or related comment. You can specify output filters to limit the list to a specific location. Only responses matching the filter criteria are selected.

The format of the output depends on the value specified in the ALCMP_RQ_RECTYP and APCMP_RQ_BLDSCCL fields. Refer to the description of these fields for additional information. Partial or full wildcarding is permitted on all the inventory location and output filter fields. Leaving a field value blank is equivalent to specifying an '*'.

By fully qualify the location data, you are requesting a list for a specific version of the element, member, object or comment. In most cases, however, you may find the results to be more meaningful if all the location data fields, except for the element, member, object or comment name are left blank or set to '*' (wildcard).

Assembler: ENHALCMP

COBOL: ECHALCMP

2.39.1 ALCMP_RQ Request Structure Fields

Immediately following the header is the data area of the ALCMP_RQ request structure. The request structure is where you set your selection criteria. All request selection fields are explained in the following table.

Field	Length	Description
ALCMP_RQ_RQTYPE	Character 1	Type of entity the request is for: E - Footprinted element M - Non-footprinted member O - Related object C - Related comment

2.39.1.1 Element Location Data – Request Type (E)

Field	Length	Description
ALCMP_RQ_ELM	Character 10	Element you wish to obtain relationship data for. The inventory location data is applied to the element name.

Field	Length	Description
ALCMP_RQ_ENV	Character 8	Environment name
ALCMP_RQ_SYSTEM	Character 8	System name
ALCMP_RQ_SUBSYS	Character 8	Subsystem name
ALCMP_RQ_TYPE	Character 8	Type name
ALCMP_RQ_STG_NUM	Character 1	Stage number (1/2)
ALCMP_RQ_DIR	Character 1	Direction indicator: C - Component list request for an element W - Where used list request for a component
	Character 26	** Reserved field **

2.39.1.2 Member Location Data — Request Type (M)

Field	Length	Description
ALCMP_RQ_MBR	Character 10	Member you want to obtain relationship data for.
ALCMP_RQ_DSN	Character 44	Data set name
	Character 16	** Reserved field **

2.39.1.3 Object Location Data — Request Type (O)

Field	Length	Description
ALCMP_RQ_OBJNAME	Character 70	Related object you want to obtain relationship data for.

2.39.1.4 Comment Location Data — Request Type (C)

Field	Length	Description
ALCMP_RQ_COMMNAME	Character 70	Related comment you want to obtain relationship data for.

2.39.1.5 Output Filters — Request Type (C, E, M, O)

Field	Length	Description
ALCMP_RQ_EXCLREL	Character 1	Exclude related element/member data (Y/N)
ALCMP_RQ_FENV	Character 8	Environment name
ALCMP_RQ_FSYSTEM	Character 8	System name
ALCMP_RQ_FSUBSYS	Character 8	Subsystem name
ALCMP_RQ_FTYPE	Character 8	Type name
ALCMP_RQ_FSTG_NUM	Character 1	Stage number (1/2)

2.39.1.6 Build Generate Action SCL

Field	Length	Description
ALCMP_RQ_BLDSCS	Character 1	Build GENERATE action SCL statements (Y/N). If you code a value of Y in this field, the output consists of 80 character SCL statements. If you code a value of N, standard API response records, mapped by ALCMP_RS, are returned. When selecting this option, you must define AACTL_LIST_DDN as a fixed block, 80 character file.
ALCMP_RQ_GENV	Character 8	Environment name
ALCMP_RQ_GSYSTEM	Character 8	System name
ALCMP_RQ_GSUBSYS	Character 8	Subsystem name
ALCMP_RQ_GTYPE	Character 8	Type name
ALCMP_RQ_GSTG_ID	Character 1	Stage id as defined in the C1DEFLTS table. You can code stage id or stage number, but not both.
ALCMP_RQ_GSTG_NUM	Character 1	Stage number (1/2). You can code stage id or stage number, but not both.
ALCMP_RQ_GCCID	Character 12	CCID action option
ALCMP_RQ_GCOMMENT	Character 40	Comment action option

Field	Length	Description
ALCMP_RQ_GCOPYBACK	Character 1	Copyback element action option (Y/N)
ALCMP_RQ_GSEARCH	Character 1	Search map action option (Y/N)
ALCMP_RQ_GOVESIGNO	Character 1	Override signout action option (Y/N)
ALCMP_RQ_GPROGRO	Character 8	Processor group name
	Character 5	** Reserved field **

Note: If you code a value of N in the ALCMP_RQ_BLDSCSCL field, all the ALCMP_RQ_G fields are ignored. If you code a value of Y, the values specified in these fields appear in the GENERATE action SCL statements.

2.39.2 ALCMP_RS Response Structure Fields

This response structure defines the layout of the data returned by the API. If you code a value of Y in the ALCMP_RQ_BLDSCSCL field, 80 character GENERATE action SCL statements are returned instead of the data shown below. A variable, ALCMP_RS_SCLSTMT, defined as an 80 character field, exists to allow you to reference the SCL statement records.

Field	Length	Description
ALCMP_RS_RECTYP	Character 1	Type of entity this response is: 1 - Footprinted element 2 - Non-footprinted member 3 - Related element 4 - Related member 5 - Related object 6 - Related comment

Field	Length	Description
ALCMP_RS_LEVEL	Character 2	<p>Possible values are 01, 02 or 03. Level 1 is always the record that matches the entity (element, member, comment or object) you specified in the request structure. If occurrences of the requested entity are found at more than one location, then multiple sets of records are returned. The meaning of the values vary, depending on the request type (E, M, O or C) and direction (C or W) you specify.</p> <p>List component for element - (01) Element, (02) Component</p> <p>List where-used for element - (01) Element, (02) Parent element, (03) Parent element of level 2 element</p> <p>List where-used for member - (01) Member, (02) Parent element</p> <p>List where-used for comment or object - (01) Comment/Object, (02) Parent element</p>
	Character 1	** Reserved field **

2.39.2.1 RECTYP 1 or 3 Format

Field	Length	Footprinted Element Data
ALCMP_RS_ELM	Character 10	Element name
ALCMP_RS_TYPE	Character 8	Type name
ALCMP_RS_ENV	Character 8	Environment name
ALCMP_RS_SYSTEM	Character 8	System name
ALCMP_RS_SUBSYS	Character 8	Subsystem name
ALCMP_RS_STG_NUM	Character 1	Stage number. Possible values are 1 or 2
ALCMP_RS_STG_ID	Character 1	Stage id as defined in the C1DEFLTS table
ALCMP_RS_STG_NAME	Character 8	Stage name as defined in the C1DEFLTS table

Field	Length	Footprinted Element Data
	Character 32	** Reserved field **

2.39.2.2 RECTYP 2 or 4 Format

Field	Length	Non-Footprinted Member Data
ALCMP_RS_NFPDSN	Character 44	Non-FP data set name
ALCMP_RS_NFPMBR	Character 10	Non-FP member name
	Character 30	** Reserved field **

2.39.2.3 RECTYP 5 or 6 Format

Field	Length	Related Object or Comment Data
ALCMP_RS_RELTEXTL	2 bytes Binary	Length of object name or comment data
ALCMP_RS_RELTEXT	Character 70	Object or comment data
	Character 12	** Reserved field **

2.40 List Data Set

The list data set API function call extracts information about data sets from the MCF that satisfies the criteria you define in the ALDSN_RQ request structure. Depending on the response file DDN (AACTL_LIST_DDND) field value, the first of the last response structure is placed in your defined response area, ALDSN_RS. Refer to 2.2, “Control Structure” on page 2-3 for details. This allows you to check the response quickly if you are looking for a specific data set. The API also writes all responses generated by your request to an external data set if you specified a file output DD name in the control structure.

Assembler: ENHALDSN

COBOL: ECHALDSN

2.40.1 ALDSN_RQ Request Structure Fields

Immediately following the header is the data area of the ALDSN_RQ request structure. The request structure is where you set your selection criteria. All request selection fields are explained in the following table.

Field	Length	Description
ALDSN_RQ_RETURN	Character 1	F - for return only the first record that satisfies the request. A - for return all records that satisfy the request.
ALDSN_RQ_ENV	Character 8	Environment name. This field cannot contain a wildcard character.
ALDSN_RQ_SYSTEM	Character 8	System name. This field cannot contain a wildcard character.
ALDSN_RQ_STG_ID	Character 1	Stage id. This field cannot contain a wildcard character.
ALDSN_RQ_DSNID	Character 2	Data Set id. This field can contain a wildcard character.

Searching: Map searching is not available for this request. All location values (Environment, System, and Stage id) must be explicitly specified. The data set id value may contain a wildcard character.

2.40.2 ALDSN_RS Response Structure Fields

Immediately following its header is the data area of the ALDSN_RS response structure. The information contained in the response structure is explained in the following tables.

Field	Length	Description
ALDSN_RS_SITE	Character 1	Site id
ALDSN_RS_ENV	Character 8	Environment name
ALDSN_RS_SYSTEM	Character 8	System name
ALDSN_RS_STG_NAME	Character 8	Stage name
ALDSN_RS_STG_ID	Character 1	Stage id
ALDSN_RS_STG_NUM	Character 1	Stage number (1/2)
ALDSN_RS_DSNID	Character 2	Data set record id
ALDSN_RS_DSNTY	Character 2	Data set type: PO, PV, LB, EL, VK
ALDSN_RS_DSN	Character 44	Data set name
ALDSN_RS_UPD_DATE	Zoned Char 8	Record update date YYYYMMDD
ALDSN_RS_UPD_TIME	Zoned Char 8	Record update time HHMMSSSTT
ALDSN_RS_UPD_CNT	Zoned Char 8	Record update count
ALDSN_RS_UPD_USER	Character 8	Update user id

2.41 List Directory

The list directory API function call allows you to build either a directory list of a file or a list of CSECT for one or more members of a load library. Specify a DDN or DSN along with a member and optional through member name. The API returns a list of members found in that file, along with any footprint information. Wildcarding is permitted on the member and through member name fields.

This function supports HFS file structures. No fields in the function call are required when processing an HFS file structure, but you must specify an initialized copy of this function as a parameter when calling the API. You must specify the path and file names in the Request Extension block (ENHAAREB) for this type of request. Wildcarding is permitted on the file name field.

Assembler: ENHALDIR

COBOL: ECHALDIR

2.41.1 ALDIR_RQ Request Structure Fields

Immediately following the header is the data area of the ALDIR_RQ request structure. The request structure is where you set your selection criteria. All request selection fields are explained in the following table.

Field	Length	Description
ALDIR_RQ_DDN	Character 8	File or DD name where member(s) reside. You must specify either DDN or DSN, but not both.
ALDIR_RQ_DSN	Character 44	Data set name where member(s) reside. You must specify either DDN or DSN, but not both.
ALDIR_RQ_MBR	Character 10	Member name. Wildcarding is permitted.
ALDIR_RQ_MBR_THRU	Character 10	Through member name. Wildcarding is permitted.

Field	Length	Description
ALDIR_RQ_CSECT	Character 1	Request CSECT flag. If specified, the API builds a list of CSECTs for the member(s) of the load library specified. To activate this feature, code a value of 'Y' in this field. If you specify this option the returned count (AACTL_#RETURNED) contains the number of members processed and the selected count (AACTL_#SELECTED) contains the total number of CSECTs found in the returned members.
ALDIR_RQ_SUBDIR	Character 1	This option is for HFS processing. By default you only receive a list of files that exist within the directory you specified. If you want to include all the sub-directories along with the files in the list, code a value of 'Y' in this field. The ALDIR_RS_MODE field in the response structure indicates the type associated with each entry.
	Character 2	** Reserved field **

2.41.2 ALDIR_RS Response Structure Fields

This response structure defines the layout of the data returned by the API. The CSECT field is blank unless you specify the CSECT option in the request structure and the input file is a load library. Endeavor footprint data only appears for members within Endeavor controlled libraries.

Field	Length	Endeavor Footprint Data
ALDIR_RS_ENV	Character 8	Environment name
ALDIR_RS_SYSTEM	Character 8	System name
ALDIR_RS_SUBSYS	Character 8	Subsystem name
ALDIR_RS_ELEMENT	Character 10	Element name

Field	Length	Endevor Footprint Data
ALDIR_RS_TYPE	Character 8	Type name
ALDIR_RS_STG_NUM	Character 1	Stage number. Possible values are 1 or 2.
ALDIR_RS_VVLL	Character 5	Version and level. Format is VV.LL (01.99).
ALDIR_RS_DATE	Character 7	Footprint date. Format is DDMMYY (31JAN01).
ALDIR_RS_TIME	Character 5	Footprint time. Format is HH:MM (23:59).
ALDIR_RS_SITEID	Character 1	Site id
ALDIR_RS_LD_FLAG	Character 1	Load utility flag. Possible values are 'Y' or blank.
ALDIR_RS_CSECT	Character 8	CSECT name. This field is blank unless you specify the CSECT option in the request structure and the input file is a load library.
ALDIR_RS_MODE	Character 1	HFS file mode. Used in conjunction with HFS files. Indicates if this entry is a directory name or an HFS file name. Possible values are: D - directory F - file U - unknown
	Character 1	** Reserved field **
ALDIR_RS_MBRL	Binary (2 bytes)	Length of member or HFS file name. Possible values are 1-255.
ALDIR_RS_MBR	Character 255	Member or HFS file name
	Character 3	** Reserved field **

2.42 List Environment

The list environment API function call extracts information about environments in Endeavor that satisfies the criteria you define in the ALENV_RQ request structure. Depending on the response file DDN (AACTL_LIST_DDN) field value, the first of the last response structure is placed in your defined response area, ALENV_RS. Refer to 2.2, “Control Structure” on page 2-3 for details. This allows you to check the response quickly if you are looking for a specific environment. The API also writes all responses generated by your request to an external data set if you specified a file output DD name in the control structure.

Assembler: ENHALENV

COBOL: ECHALENV

2.42.1 ALENV_RQ Request Structure Fields

Immediately following the header is the data area of the ALENV_RQ request structure. The request structure is where you set your selection criteria. All request selection fields are explained in the following table.

Field	Length	Description
ALENV_RQ_PATH	Character 1	Mapping path: 'L' for Logical 'P' for Physical
ALENV_RQ_RETURN	Character 1	'F' for return only the first record that satisfies the request. 'A' for return all records that satisfy the request.
ALENV_RQ_SEARCH	Character 1	Mapping argument: 'A' for Search All the way up the map. 'B' for Search Between the two specified environments. 'N' for No Search. 'E' for Search nExt specified environment then up the map. 'R' for Search the Range, between and including the specified environments.

Field	Length	Description
ALENV_RQ_ENV	Character 8	Environment name. This field cannot contain a wildcard character.
ALENV_RQ_TOENV	Character 8	To environment name. If specified, this field cannot contain a wildcard character. Optional.

Searching: If you specify:

- 'E', 'B', or 'R' for the ALENV_RQ_SEARCH search argument, you cannot use a wildcard in the Environment name.
- 'B', or 'R' for the ALENV_RQ_SEARCH search argument, you must specify the To Environment in the ALENV_RQ_TOENV field. You cannot use a wildcard.
- The To Environment in the ALENV_RQ_TOENV field, it will be ignored unless you also specify the 'B' or 'R' search option.

2.42.2 ALENV_RS Response Structure Fields

Immediately following its header is the data area of the ALENV_RS response structure. The information contained in the response structure is explained in the following table.

Field	Length	Description
ALENV_RS_SITE	Character 1	Site id
ALENV_RS_ENV	Character 8	Environment id
ALENV_RS_TITLE	Character 40	Description
ALENV_RS_USEC	Character 8	User security name table
ALENV_RS_RSEC	Character 8	Resource security name table
ALENV_RS_SMFSEC	Character 1	SMF Recording - Security. Value Y for Yes or N for No.
ALENV_RS_SMFACT	Character 1	SMF Recording - Actions. Value Y for Yes or N for No.
ALENV_RS_SMFENV	Character 1	SMF Recording - Environment. Value Y for Yes or N for No.
ALENV_RS_DBAVL	Character 1	DB Bridge Available. Value Y for Yes or N for No.
ALENV_RS_DBACT	Character 1	DB Bridge Active. Value Y for Yes or N for No.
ALENV_RS_DBOPT1	Character 1	DB Bridge Option 1. Value Y for Yes or N for No.

Field	Length	Description
ALENV_RS_DBOPT2	Character 1	DB Bridge Option 2. Value Y for Yes or N for No.
ALENV_RS_DBOPT3	Character 1	DB Bridge Option 3. Value Y for Yes or N for No.
ALENV_RS_DBOPT4	Character 1	DB Bridge Option 4. Value Y for Yes or N for No.

2.43 List Processor Group

The list processor group API function call extracts information from the MCF that satisfies the criteria you define in the ALPGR_RQ request structure about the processor group. It will also extract any symbolic overrides that have been defined for any of the group's processors.

A record is produced for each processor within a group. Processor group information that relates to the whole group is replicated on each processor record. Also, if overrides exist for any processor, the whole record is replicated with an override value appended to the record. For example, if a group has two processors defined, and the first processor has two overrides while the second only has one, three records are produced:

- The first record contains the group's information. The first processor information is followed by the first symbolic override data.
- The second record contains the same group and processor information followed by the second override data.
- The third record contains the same group information, the second processor information, followed by the symbolic override data pertaining to the second processor.

Depending on the response file DDN (AACTL_LIST_DDND) field value, the first of the last response structure is placed in your defined response area, ALPGR_RS. Refer to 2.2, "Control Structure" on page 2-3 for details. This allows you to check the response quickly if you are looking for a specific processor group. The API also writes all responses generated by your request to an external data set if you specified a file output DD name in the control structure.

Assembler: ENHALPGR

COBOL: ECHALPGR

2.43.1 ALPGR_RQ Request Structure Fields

Immediately following the header is the data area of the ALPGR_RQ request structure. The request structure is where you set your selection criteria. All request selection fields are explained in the following table.

Field	Length	Description
ALPGR_RQ_RETURN	Character 1	'F' for return only the first record that satisfies the request. 'A' for return all records that satisfy the request.

Field	Length	Description
ALPGR_RQ_ENV	Character 8	Environment name. This field cannot contain a wildcard character.
ALPGR_RQ_SYSTEM	Character 8	System name. This field cannot contain a wildcard character.
ALPGR_RQ_TYPE	Character 8	Type name. This field cannot contain a wildcard character.
ALPGR_RQ_STG_ID	Character 1	Stage id. This field cannot contain a wildcard character.
ALPGR_RQ_PGRP	Character 8	Processor Group Name. This field can contain a wildcard character.

Searching: Map searching is not available for this request. All location values (Environment, System, Type, and Stage id) must be explicitly specified. The processor group name can contain a wildcard.

2.43.2 ALPGR_RS Response Structure Fields

Immediately following its header is the data area of the ALPGR_RS response structure. The information contained in the response structure is explained in the following tables.

Field	Length	Description
ALPGR_RS_SITE	Character 1	Site id
ALPGR_RS_ENV	Character 8	Environment name
ALPGR_RS_SYSTEM	Character 8	System name
ALPGR_RS_TYPE	Character 8	Type name
ALPGR_RS_STG_NAME	Character 8	Stage name
ALPGR_RS_STG_ID	Character 1	Stage id
ALPGR_RS_STG_NUM	Character 1	Stage number (1/2)
ALPGR_RS_PGRP	Character 8	Processor group name
ALPGR_RS_PROTY	Character 4	Processor type: DEL - for delete processor GEN - for generate processor MOVE - for move processor

Field	Length	Description
ALPGR_RS_SYM# ALPGR-RS-SYMNUM (COBOL)	Zoned Char 4	Symbolic override number. If no symbolic overrides exist for this processor, this value will be set to zero.
ALPGR_RS_UPD_DATE	Zoned Char 8	Record update date YYYYMMDD
ALPGR_RS_UPD_TIME	Zoned Char 8	Record update time HHMMSSTT
ALPGR_RS_UPD_USER	Character 8	Update User id
ALPGR_RS_DESC	Character 50	Processor group description
ALPGR_RS_NEXT_PGR	Character 8	Next pathed processor group
ALPGR_RS_PGRTYPE	Character	Output type
ALPGR_RS_PMOVE	Character 1	Processor to use on Move actions: G - for Generate M - for Move
ALPGR_RS_PROFG	Character 1	Foreground Processing flag. Value Y for Yes and N for No.
ALPGR_RS_PRONME	Character 8	Processor name
ALPGR_RS_PXFER	Character 1	Processor to use on Transfer actions: G - for Generate M - for Move
ALPGR_RS_SYM_LEN	Zoned Char 2	Override symbol length
ALPGR_RS_SYM	Character 8	Override symbol
ALPGR_RS_SVAL_LEN	Zoned Char 4	Override symbol value length
ALPGR_RS_SVAL	Character 256	Override symbol value

2.44 List Site

The list site API function call extracts information about sites. Depending on the response file DDN (AACTL_LIST_DDN) field value, the first of the last response structure is placed in your defined response area, ALSIT_RS. Refer to 2.2, “Control Structure” on page 2-3 for details. This allows you to check the response quickly if you are looking for specific site information. The API also writes this data to an external data set if you specified a file output DD name in the control structure.

Assembler: ENHALSIT

COBOL: ECHALSIT

2.44.1 ALSIT_RQ Request Structure Fields

The ALSIT_RQ request structure defines your List Site fields, which in this case is only the Site header. No additional data can be specified.

2.44.2 ALSIT_RS Response Structure Fields

Immediately following the header is the data area of the ALSIT_RS response structure. The information contained in the response structure is explained in the following table.

Field	Length	Description
ALSIT_RS_NDVRREL	Character 6	Endevor release identifier.
ALSIT_RS_SITEID	Character 1	Site id
ALSIT_RS_SITENAME	Character 50	Site name
ALSIT_RS_DFLTDATA	Character 8	C1DEFLTS assembly date
ALSIT_RS_DFLTTIME	Character 8	C1DEFLTS assembly time
ALSIT_RS_APPREQD	Character 1	Approvals required
ALSIT_RS_EXITAUTH	Character 8	Authorized library for exits
ALSIT_RS_ELNKXTBL	Character 8	ELink data stream translate table
ALSIT_RS_LINESPP	Zoned Char 2	Lines per page (reports and logs)
ALSIT_RS_MACLIB	Character 44	Endevor installation macro library
ALSIT_RS_PKGSEC	Character 1	Perform Security Check at Cast flag. Value Y for Yes and N for No.

Field	Length	Description
ALSIT_RS_PKGCVL	Character 1	Component Validation flag O - for Optional Y - for required
ALSIT_RS_PKGTSO	Character 1	Package Execution Valid in Foreground flag. Value Y for Yes and N for No.
ALSIT_RS_MFMTCCID	Character 1	Mixed Case for CCID flag. Value Y for Yes and N for No.
ALSIT_RS_MFMTCMNT	Character 1	Mixed Case for Comment flag. Value Y for Yes and N for No.
ALSIT_RS_MFMTDESC	Character 1	Mixed Case for Description flag. Value Y for Yes and N for No.
ALSIT_RS_PKGDSN	Character 44	Endevor Package data set name
ALSIT_RS_CCIDVAL	Character 44	Endevor CCID Validation table
ALSIT_RS_MODHLI	Character 8	High Level Qualifier for DISP=MOD temporary data set names on processor executions.
ALSIT_RS_ACCSTABL	Character 8	ESI access security table
ALSIT_RS_ACMOPT	Character 1	Endevor ACM Option Available flag. Value Y for Yes and N for No.
ALSIT_RS_DB2OPT	Character 1	Endevor DB2 Option Available flag. Value Y for Yes and N for No.
ALSIT_RS_ELKOPT	Character 1	Endevor ELink Option Available flag. Value Y for Yes and N for No.
ALSIT_RS_ESIOPT	Character 1	Endevor ESI Option Available flag. Value Y for Yes and N for No.
ALSIT_RS_INFOP	Character 1	Endevor INFOMAN Option Available flag. Value Y for Yes and N for No.
ALSIT_RS_JRNLGRP	Character 14	Point-in-time recovery journal group.
ALSIT_RS_LPVOPT	Character 1	Endevor LIB/PNV Option Available flag. Value Y for Yes and N for No.

Field	Length	Description
ALSIT_RS_PDMOPT	Character 1	Endevor PDM Option Available flag. Value Y for Yes and N for No.
ALSIT_RS_PRCOPT	Character 1	Endevor Processor Option Available. Value Y for Yes and N for No.
ALSIT_RS_QEDOPT	Character 1	Endevor Quick Edit Option Available flag. Value Y for Yes and N for No.
ALSIT_RS_SOFETCH	Character 1	Signout Source on Fetch (Retrieve Action) Option flag. Value Y for Yes and N for No.
ALSIT_RS_GNIPSOUT	Character 1	Signout on Generate in Place Action flag. Value Y for Yes and N for No.
ALSIT_RS_SMFREC# ALSIT-RS-SMFRECNUM (COBOL)	Zoned Char 3	SMF record number
ALSIT_RS_RACFUID	Character 8	Endevor alternate RACF userid
ALSIT_RS_RACFGRP	Character 8	Endevor alternate RACF group
ALSIT_RS_RACFPWD	Character 8	Endevor alternate RACF password
ALSIT_RS_LIBENV	Character 2	CA-Panvalet/CA-Librarian (PV/LB) environment
ALSIT_RS_LIBPGM	Character 8	CA-Librarian interface program name
ALSIT_RS_RJCLROOT	Character 4	Package ship remote JCL root model member
ALSIT_RS_SPFEDIT	Character 8	Reserve QNAME for non-load libraries
ALSIT_RS_SYSIEWLP	Character 8	Reserve QNAME for load libraries
ALSIT_RS_TSOE	Character 1	TSO-E Installed flag. Value Y for Yes and N for No.
ALSIT_RS_WRKUNIT	Character 8	Esoteric non-vio work unit name.
ALSIT_RS_WRKVOL	Character 6	Work unit VOLSER

Field	Length	Description
ALSIT_RS_BATCHID	Zoned Char 1	Batch Id flag: 0 - Userid taken from jobname 1 - Userid taken from user= parameter 2 - Userid taken from user= parameter, but if blank, take from jobname
ALSIT_RS_UIDLOCO	Zoned Char 1	Userid offset in jobcard
ALSIT_RS_UIDLOCL	Zoned Char 1	Userid length in jobname
ALSIT_RS_VIOUNIT	Character 8	Esoteric VIO unit name

2.45 List Stage

The list stage API function call extracts information about stages in Endeavor that satisfies the criteria you define in the ALSTG_RQ request structure. Depending on the response file DDN (AACTL_LIST_DDN) field value, the first of the last response structure is placed in your defined response area, ALSTG_RS. Refer to 2.2, “Control Structure” on page 2-3 for details. This allows you to check the response quickly if you are looking for a specific stage. The API also writes all responses generated by your request to an external data set if you specified a file output DD name in the control structure.

Assembler: ENHALSTG

COBOL: ECHALSTG

2.45.1 ALSTG_RQ Request Structure Fields

Immediately following the header is the data area of the ALSTG_RQ request structure. The request structure is where you set your selection criteria. All request selection fields are explained in the following table.

Field	Length	Description
ALSTG_RQ_PATH	Character 1	Mapping path: 'L' for Logical 'P' for Physical
ALSTG_RQ_RETURN	Character 1	'F' for return only the first record that satisfies the request. 'A' for return All
ALSTG_RQ_SEARCH	Character 1	Mapping argument: 'A' for Search All the way up the map. 'B' for Search Between the two specified environments and stages. 'N' for No Search. 'E' for Search nExt specified environment/stage then up the map. 'R' for Search the Range, between and including the specified environments and stages.
ALSTG_RQ_ENV	Character 8	Environment name. This field cannot contain a wildcard character.

Field	Length	Description
ALSTG_RQ_TOENV	Character 8	To Environment name. If specified, this field cannot contain a wildcard character. Optional.
ALSTG_RQ_STG_ID	Character 1	Stage id. This field can be a wildcard character.
ALSTG_RQ_TOSTG_ID	Character 1	To Stage id. If specified, this field cannot contain a wildcard character. Optional.

Searching: If you specify 'B', or 'R' for the ALSTG_RQ_SEARCH search argument, you must specify the To Environment in the ALSTG_RQ_TOENV field and the To Stage Id in the ALSTG_RQ_TOSTG field. You cannot use a wildcard. In addition, these fields will be ignored unless the 'B' or 'R' search field is also specified.

2.45.2 ALSTG_RS Response Structure Fields

Immediately following its header is the data area of the ALSTG_RS response structure. The information contained in the response structure is explained in the following table.

Field	Length	Description
ALSTG_RS_SITE	Character 1	Site id
ALSTG_RS_ENV	Character 8	Environment name
ALSTG_RS_STG_NAME	Character 8	Stage name
ALSTG_RS_STG_ID	Character 1	Stage id
ALSTG_RS_STG_NUM	Character 1	Stage number
ALSTG_RS_TITLE	Character 20	Title
ALSTG_RS_DSN	Character 44	MCF data set name

2.46 List Subsystem

The list subsystem API function call extracts information about subsystems from the MCF that satisfies the criteria you define in the ALSBS_RQ request structure. Depending on the response file DDN (AACTL_LIST_DDN) field value, the first of the last response structure is placed in your defined response area, ALSBS_RS. Refer to 2.2, “Control Structure” on page 2-3 for details. This allows you to check the response quickly if you are looking for a specific subsystem. The API also writes all responses generated by your request to an external data set if you specified a file output DD name in the control structure.

Assembler: ENHALSBS

COBOL: ECHALSBS

2.46.1 ALSBS_RQ Request Structure Fields

Immediately following the header is the data area of the ALSBS_RQ request structure. The request structure is where you set your selection criteria. All request selection fields are explained in the following table.

Field	Length	Description
ALSBS_RQ_PATH	Character 1	Mapping path: 'L' for Logical 'P' for Physical
ALSBS_RQ_RETURN	Character 1	'F' for return only the first record that satisfies the request. 'A' for return all records that satisfy the request.
ALSBS_RQ_SEARCH	Character 1	Mapping argument: 'A' for Search All the way up the map. 'B' for Search Between the two specified environments and stages. 'N' for No Search. 'E' for Search nExt specified environment/stage then up the map. 'R' for Search the Range, between and including the specified environments and stages.

Field	Length	Description
ALSBS_RQ_ENV	Character 8	Environment name. This field cannot contain a wildcard character.
ALSBS_RQ_TOENV	Character 8	To Environment name. If specified, this field cannot contain a wildcard character. Optional.
ALSBS_RQ_STG_ID	Character 1	Stage id. This field can contain a wildcard character.
ALSBS_RQ_TOSTG_ID	Character 1	To Stage id. If specified, this field cannot contain a wildcard character. Optional.
ALSBS_RQ_SYSTEM	Character 8	System name. This field can contain a wildcard character.
ALSBS_RQ_SUBSYS	Character 8	Subsystem name. This field can contain a wildcard character.

Searching: If you specify 'B', or 'R' for the ALSBS_RQ_SEARCH search argument, you must specify the To Environment in the ALSBS_RQ_TOENV field and the To Stage Id in the ALSBS_RQ_TOSTG_ID field. You cannot use a wildcard. In addition, these fields will be ignored unless you also specify the 'B' or 'R' search field.

2.46.2 ALSBS_RS Response Structure Fields

Immediately following its header is the data area of the ALSBS_RS response structure. The information contained in the response structure is explained in the following table.

Field	Length	Description
ALSBS_RS_SITE	Character 1	Site id
ALSBS_RS_ENV	Character 8	Environment name
ALSBS_RS_SYSTEM	Character 8	System name
ALSBS_RS_SUBSYS	Character 8	Subsystem name
ALSBS_RS_STG_NAME	Character 8	Stage name
ALSBS_RS_STG_ID	Character 1	Stage id
ALSBS_RS_STG_REL	Zoned Char 4	Relative mapped stage number
ALSBS_RS_UPD_CNT	Zoned Char 8	Record update count
ALSBS_RS_UPD_DATE	Zoned Char 8	Update date YYYYMMDD
ALSBS_RS_UPD_TIME	Zoned Char 8	Update time HHMMSSSTT
ALSBS_RS_UPD_USER	Character 8	Update user id

2.46 List Subsystem

Field	Length	Description
ALSBS_RS_TITLE	Character 50	Subsystem title
ALSBS_RS_NXT_SBS	Character 8	Next subsystem name in path
ALSBS_RS_FMID	Zoned Char 5	Record created release id

2.47 List System

The list system API function call extracts information about systems from MCF files that satisfies the criteria you define in the ALSYS_RQ request structure. Depending on the response file DDN (AACTL_LIST_DDND) field value, the first of the last response structure is placed in your defined response area, ALSYS_RS. Refer to 2.2, “Control Structure” on page 2-3 for details. This allows you to check the response quickly if you are looking for a specific system. The API also writes all responses generated by your request to an external data set if you specified a file output DD name in the control structure.

Assembler: ENHALSYS

COBOL: ECHALSYS

2.47.1 ALSYS_RQ Request Structure Fields

Immediately following the header is the data area of the ALSYS_RQ request structure. The request structure is where you set your selection criteria. All request selection fields are explained in the following table:

Field	Length	Description
ALSYS_RQ_PATH	Character 1	Mapping path: 'L' for Logical 'P' for Physical
ALSYS_RQ_RETURN	Character 1	'F' for return only the first record that satisfies the request. 'A' for return all records that satisfy the request.
ALSYS_RQ_SEARCH	Character 1	Mapping argument: 'A' for Search All the way up the map. 'B' for Search Between the two specified environments and stages. 'N' for No Search. 'E' for Search nExt specified environment/stage then up the map. 'R' for Search the Range, between and including the specified environments and stages.

Field	Length	Description
ALSYS_RQ_ENV	Character 8	Environment name. You cannot use a wildcard character in this field.
ALSYS_RQ_TOENV	Character 8	To Environment name. If specified, it cannot contain a wildcard character. Optional.
ALSYS_RQ_STG_ID	Character 1	Stage id. You can use a wildcard character in this field.
ALSYS_RQ_TOSTG_ID	Character 1	To Stage id. If specified, it cannot contain a wildcard character. Optional.
ALSYS_RQ_SYSTEM	Character 8	System name. You can use a wildcard character in this field.

Searching: If you specify 'B', or 'R' for the ALSYS_RQ_SEARCH search argument, you must specify the To Environment in the ALSYS_RQ_TOENV field and the To Stage Id in the ALSYS_RQ_TOSTG_ID field. You cannot use a wildcard. In addition, these fields will be ignored unless you also specify the 'B' or 'R' search field.

2.47.2 ALSYS_RS Response Structure Fields

Immediately following its header is the data area of the ALSYS_RS response structure. The information contained in the response structure is explained in the following table.

Field	Length	Description
ALSYS_RS_SITE	Character 1	Site id
ALSYS_RS_ENV	Character 8	Environment name
ALSYS_RS_SYSTEM	Character 8	System name
ALSYS_RS_STG_NAME	Character 8	Stage name
ALSYS_RS_STG_ID	Character 1	Stage id
ALSYS_RS_STG_REL	Zoned Char 4	Relative mapped stage number
ALSYS_RS_UPD_CNT	Zoned Char 8	Record update count
ALSYS_RS_UPD_DATE	Zoned Char 8	Update date YYYYMMDD
ALSYS_RS_UPD_TIME	Zoned Char 8	Update time HHMMSSSTT
ALSYS_RS_UPD_USER	Character 8	Update user id
ALSYS_RS_TITLE	Character 50	System title
ALSYS_RS_NXT_SYS	Character 8	Next system name in path

Field	Length	Description
ALSYS_RS_LOADLIB	Character 44	Processor load library
ALSYS_RS_LISTING	Character 44	Processor listing library
ALSYS_RS_COMMENT	Character 1	Comment Required flag. Value is Y for Yes or No for No.
ALSYS_RS_CCID	Character 1	CCID Required flag. Value is Y for Yes or No for No.
ALSYS_RS_SISO1	Character 1	Signout Required flag. Value is Y for Yes or No for No.
ALSYS_RS_SISO2	Character 1	Validate Retrieve Dataset flag. Value is Y for Yes or No for No.
ALSYS_RS_JUMP	Character 1	Jump Option Required flag. Value is Y for Yes or No for No.
ALSYS_RS_BAK_DATE	Zoned Char 8	Backup date YYYYMMDD
ALSYS_RS_BAK_TIME	Zoned Char 8	Backup Time HHMMSSTT
ALSYS_RS_FMID	Zoned Char 5	Record created release id
ALSYS_RS_DREG	Character 1	Duplicate element name registration checking feature. Values are Y for Yes or N for No.
ALSYS_RS_DREGS	Character 1	Duplicate element name registration check message severity level. Values are Y for Yes or N for No.
ALSYS_RS_PREG	Character 1	Duplicate processor output type registration checking feature. Values are Y for Yes or N for No.
ALSYS_RS_PREGS	Character 1	Duplicate processor output registration check message severity level. Acceptable values are blank, W, C or E.

2.48 List Type

The list type API function call extracts information about types from the MCF that satisfies the criteria you define in the `ALTYP_RQ` request structure. Depending on the response file DDN (`AACTL_LIST_DDN`) field value, the first of the last response structure is placed in your defined response area, `ALTYP_RS`. Refer to 2.2, “Control Structure” on page 2-3 for details. This allows you to check the response quickly if you are looking for a specific type. The API also writes all responses generated by your request to an external data set if you specified a file output DD name in the control structure.

Assembler: `ENHALTYP`

COBOL: `ECHALTYP`

2.48.1 `ALTYP_RQ` Request Structure Fields

Immediately following the header is the data area of the `ALTYP_RQ` request structure. The request structure is where you set your selection criteria. All request selection fields are explained in the following table:

Field	Length	Description
<code>ALTYP_RQ_PATH</code>	Character 1	Mapping path: 'L' for Logical 'P' for Physical
<code>ALTYP_RQ_RETURN</code>	Character 1	'F' for return only the first record that satisfies the request. 'A' for return all records that satisfy the request.
<code>ALTYP_RQ_SEARCH</code>	Character 1	Mapping argument: 'A' for Search All the way up the map. 'B' for Search Between the two specified environments and stages. 'N' for No Search 'E' for Search nExt specified environment/stage then up the map. 'R' for Search the Range, between and including the specified environments and stages.

Field	Length	Description
ALTYP_RQ_ENV	Character 8	Environment name. You cannot use a wildcard character in this field.
ALTYP_RQ_TOENV	Character 8	To Environment name. If specified, this field cannot contain a wildcard character. Optional.
ALTYP_RQ_STG_ID	Character 1	Stage id. This field can contain a wildcard character.
ALTYP_RQ_TOSTG_ID	Character 1	To Stage id. If specified, this field cannot contain a wildcard character. Optional.
ALTYP_RQ_SYSTEM	Character 8	System name. This field can contain a wildcard character.
ALTYP_RQ_TYPE	Character 8	Type name. This field can contain a wildcard character.

Searching: If you specify 'B', or 'R' for the ALTYP_RQ_SEARCH search argument, you must specify the To Environment in the ALTYP_RQ_TOENV field and the To Stage Id in the ALTYP_RQ_TOSTG_ID field. You cannot use a wildcard. In addition, these fields will be ignored unless you also specify the 'B' or 'R' search field.

2.48.2 ALTYP_RS Response Structure Fields

Immediately following its header is the data area of the ALTYP_RS response structure. The information contained in the response structure is explained in the following table.

Field	Length	Description
ALTYP_RS_SITE	Character 1	Site name
ALTYP_RS_ENV	Character 8	Environment name
ALTYP_RS_SYSTEM	Character 8	System name
ALTYP_RS_TYPE	Character 8	Type name
ALTYP_RS_TYPENBR	Character 2	Type number id
ALTYP_RS_STG_NAME	Character 8	Stage name
ALTYP_RS_STG_ID	Character 1	Stage id
ALTYP_RS_STG_NUM	Character 1	Stage number
ALTYP_RS_STG_REL	Zoned Char 4	Relative mapped stage number
ALTYP_RS_UPD_CNT	Zoned Char 8	Record update count

Field	Length	Description
ALTYP_RS_UPD_DATE	Zoned Char 8	Update date YYYYMMDD
ALTYP_RS_UPD_TIME	Zoned Char 8	Update time HHMMSSSTT
ALTYP_RS_UPD_USER	Character 8	Update user id
ALTYP_RS_FMID	Zoned Char 5	Record created release id
ALTYP_RS_NXT_TYP	Character 8	Next type name in path
ALTYP_RS_DESC	Character 50	Type description
ALTYP_RS_DPGRPNAME	Character 8	Default processor group name
ALTYP_RS_FILEEXT	Character 8	PC extension
ALTYP_RS_HOOP	Character 1	Home OPSYS: M-z/OS and OS/390 W-Workstation
ALTYP_RS_EXTLANG	Character 8	Language
ALTYP_RS_INTLANG	Character 8	PV/LB Language
ALTYP_RS_REPCT	Character 2	Regression percent
ALTYP_RS_RESEV	Character 1	Regression severity
ALTYP_RS_SLEN	Character 5	Source length
ALTYP_RS_CMPFR	Character 5	Compare from
ALTYP_RS_CMPTO	Character 5	Compare to
ALTYP_RS_AUCON	Character 1	Auto Consolidate flag. Value Y for Yes and N for No.
ALTYP_RS_AULTC	Character 3	Consolidate level
ALTYP_RS_AUCLE	Character 3	Auto consolidate level
ALTYP_RS_CAUCO	Character 1	Component Auto Consolidate flag. Value Y for Yes and N for No.
ALTYP_RS_CULTC	Character 3	Component consolidate level
ALTYP_RS_CAUCL	Character 3	Component auto consolidate level
ALTYP_RS_SOEXI	Character 1	Expand include in source output library. Value Y for Yes and N for No.
ALTYP_RS_DELTYP	Character 1	F - Forward element delta R - Reverse element delta

Field	Length	Description
ALTYP_RS_CDELTY	Character 1	F - Forward component delta R - Reverse component delta
ALTYP_RS_CMPBA	Character 1	Compress Base flag. Value Y for Yes and N for No.
ALTYP_RS_NNCR	Character 1	Not encrypted element name. Value Y for Yes and N for No.
ALTYP_RS_ODSTYP	Character 2	Source output data set type: PO / PV / LB
ALTYP_RS_ODSNME	Character 44	Source output data set name
ALTYP_RS_IDSTYP	Character 2	Include data set type: PO / PV / LB
ALTYP_RS_IDSNME	Character 44	Include data set name
ALTYP_RS_BDSTYP	Character 2	Base data set type: PO / PV / LB / EL / VK
ALTYP_RS_BDSNME	Character 44	Base data set name
ALTYP_RS_UDSTYP	Character 2	Update data set type: PO / PV / LB / EL / VK
ALTYP_RS_UDSNME	Character 44	Update data set name

Chapter 3. API Execution Reports and Trace Facilities

3.1 Overview

This chapter contains procedures and sample output for execution reports and trace facilities.

3.1.1 Execution Reports

Endevor provides execution reports that record processing information for API function calls.

3.1.2 Trace Facilities

Endevor API provides trace facilities to facilitate debugging your code. When a trace facility is executed, it writes the status and current view of the data to a response file when an API call is executed. If there is a problem completing the call, the trace facilities record the area in which the error occurred.

You should refer to trace data found in BSTERR, BSTAPI, and EN\$TRAPI DD name datasets if the return code and reason code from the API request directs you to. You should scan the files looking for error messages that might be recorded in the files. All error messages are documented in the *Error Codes and Messages Guide*. If an internal error occurs, you should send all traces to Technical Support.

Endevor API provides two trace facilities:

- The API Diagnostic Trace
- The API Internal Trace

3.2 API Execution Reports

The API writes an execution report to the message file defined by the AACTL_MSG_DDN file after each function call is processed. This report includes the requested function calls and their options and the actual processing that occurred.

Sample execution reports are provided below for the following API function calls:

- Define package action
- Element action
- Inventory list
- List action
- Update action

3.2.1 Define Package Action Function Call Sample Report

This sample report format is used to record the processing information for the API define package action request function call.

This report consists of two parts: first, the data from the request structure is formatted and printed in a format similar to the output you receive when executing an Endeavor batch request; the second part is a summary of how many records were read and selected and the highest return and reason detected.

From this report, you can see a syntax error was detected causing the action to fail.

```
(C) 2002 Computer Associates International, Inc.

                E N D E V O R   A P I   E X E C U T I O N

API0100I  DEFINE PACKAGE
API0100I  PACKAGE ID: DEFINEPACKAGE45X
API0100I  MESSAGE DDNAME: PDFMSG
API0100I  FUNCTION: M
API0100I  DESCRIPTION: PACKAGE DESCRIPTION GOES HERE012345
API0100I  TYPE PACKAGE: S
API0100I  SHARABLE: Y
API0100I  BACKOUT ENABLED: Y
API0100I  EXECUTION FROM DATE/TIME: 01AUG01 11:01 TO DATE/TIME: 30MAY02
API0100I  NOTES: 1ST LINE OF NOTES8901234567890123
API0100I  2ND LINE OF NOTES12345678901234
API0100I  3RD LINE OF NOTES1234567890123
11:34:05 API0102I  DISPATCHING API ACTION
11:34:14 API0000I  PROCESSING COMPLETE - RC=00000 REASON=00000
```

Note: To see why the define package action failed, add the DD statement, C1MSG\$1, to the JCL stream.

```
//*C1MSG$1 DD SYSOUT=*                                (DEFINE PACKAGE MESSAGES)
```

This statement contains error messages detected by the Endeavor package facility. Add it to the section with the other trace DD statements. If the error

message API0066E is returned from a define package request, the C1MSGSI DD statement provides details of why the action failed. Specify this DD statement only when debugging a define package problem.

3.2.2 Element Action Function Call Sample Report

This sample report format is used to record the processing information for all of the API element action function calls.

This report consists of two parts: first, the data from the request structure is formatted and printed in a format similar to the output you receive when executing an Endeavor batch request; the second part is the action execution log. This is the exact same output you receive when executing an Endeavor foreground or batch action.

From this report, you can determine the ADD element action request field values and determine the results of your request by viewing the execution log.

```
(C) 2002      Computer Associates International, Inc.      18JUL01 15:32:43  PAGE  1

                E N D E V O R  A P I  E X E C U T I O N  R E P O R T
15:32:43 API0000I  STARTING PRINT OF API ACTION REQUEST DATA
15:32:43 API0000I
15:32:43 API0000I  ADD ELEMENT: APIB
15:32:43 API0000I      THROUGH: APIU
15:32:43 API0000I      FROM  DSNAME: BST.BUCFR02.APISRC
15:32:43 API0000I      TO    ENVIRONMENT: INT
15:32:43 API0000I      SYSTEM: NDVRMVS
15:32:43 API0000I      SUBSYSTEM: FHB
15:32:43 API0000I      TYPE: COBCOPY
15:32:43 API0000I  OPTIONS
15:32:43 API0000I      CCID: CCIDVALUEADD
15:32:43 API0000I      COMMENT: COMMENTADD12345678901234567890
15:32:43 API0000I      UPDATE IF PRESENT: Y
15:32:43 API0000I      DELETE INPUT SOURCE: N
15:32:43 API0000I      OVERRIDE SIGNOUT: Y
15:32:43 API0000I      BYPASS GENERATE PROCESSOR: N
15:32:43 API0000I  API ACTION REQUEST DATA SUCCESSFULLY PRINTED
15:32:43 API0000I

15:32:43 C1Y0015I  STARTING PARSE OF REQUEST CARDS

STATEMENT #1
15:32:43 C1Y0016I  REQUEST CARDS SUCCESSFULLY PARSED

15:32:44 C1G0202I  ACTION #1 / STMT #1
15:32:44 C1G0203I      ADD ELEMENT APIB
15:32:44 C1G0205I      FROM DSNAME: BST.BUCFR02.APISRC  MEMBER: APIB
15:32:44 C1G0204I      TO  ENVIRONMENT: INT      SYSTEM: NDVRMVS  SUBSYSTEM
15:32:44 C1G0232I      OPTIONS:  OVERRIDE SIGNOUT, UPDATE
15:32:44 C1G0232I      CCID: CCIDVALUEADD
15:32:44 C1G0232I      COMMENT: COMMENTADD12345678901234567890
15:32:45 C1G0265I      PROCESSOR GROUP *NOPROC* FOR ELEMENT APIB WAS
15:32:46 SMGR122W      NO ELEMENT SOURCE CHANGES DETECTED
15:32:46 SMGR125I      ELEMENT APIB 01.02 NOT UPDATED BY BST.APISRC(APIB)
15:32:46 C1G0200I      REQUEST PROCESSING FOR ELM APIB COMPLETED
```

3.2.3 Inventory List Function Call Sample Report

This sample report format is used to record the processing information for all of the API inventory list function calls.

This report consists of two parts: first, the data from the request structure is formatted and printed in a format similar to the output you receive when executing an Endeavor batch request; the second part is a summary of how many records were read and selected and the highest return and reason codes detected.

From this report, you can see that three records were selected and written to a file defined to the APIEXTR DD statement and that the highest return code was zero.

```
(C) 2002      Computer Associates International, Inc.      18JUL01 13:19:12 PAGE  1
              E N D E V O R   A P I   E X E C U T I O N   R E P O R T
13:19:12  API0101I  BEGINNING OF API ACTION PROCESSING
              API0100I  LIST ENVIRONMENT
              API0100I  TO DDNAME: APIEXTR
              API0100I  ENVIRONMENT: INT
              API0100I  THRU ENVIRONMENT: PRD
              API0100I  OPTIONS
              API0100I  PATH: L RETURN: A SEARCH: A
              API0100I  API ACTION REQUEST DATA SUCCESSFULLY PRINTED
13:19:12  API0102I  DISPATCHING API ACTION
13:19:12  API0000I  RETURNED COUNT=00003, SELECTED COUNT=00003
13:19:12  API0000I  PROCESSING COMPLETE - RC=00000 REASON=00000
```

3.2.4 List Package Action Function Call Sample Reports

The following sample report formats are used to record the processing information for the API list package action function calls.

The reports consists of two parts: first, the data from the request structure is formatted and printed in a format similar to the output you receive when executing an Endeavor batch request; the second part is a summary of how many records were read and selected and the highest return and reason codes detected.

This example shows a list package header action request. There are eight packages beginning with a package id of PK1, and five were eliminated due to the selection criteria. The package header data associated with the remaining three packages was written to the data set associated with the LPKLST DD name.

3.2 API Execution Reports

```
(C) 2002      Computer Associates International, Inc.      10NOV01 10:08:09  PAGE  1

                E N D E V O R   A P I   E X E C U T I O N
10:08:09  API0101I  BEGINNING OF API ACTION PROCESSING
          API0100I  LIST PACKAGE HEADER
          API0100I  PACKAGE ID: PK1*
          API0100I  TO DDNAME: LPKLST
          API0100I  WHERE PACKAGE TYPE EQUAL: S
          API0100I  DATE TYPE CA IS OLDER THAN 30 DAYS
          API0100I  APPROVER ID: LUSOS01
          API0100I  STATUS SELECTION OPTIONS:
          API0100I  IN-EDIT.....: Y  IN-APPROVAL: Y
          API0100I  DENIED.....: N  APPROVED...: Y
          API0100I  IN_EXECUTION: Y  EXECUTED...: Y
          API0100I  COMMITTED...: Y
          API0100I
10:08:10  API0102I  DISPATCHING API ACTION
10:08:11  API0000I  RETURNED COUNT=00008, SELECTED COUNT=00003
10:08:11  API0000I  PROCESSING COMPLETE - RC=00000 REASON=00000
```

The next example contains a list package cast action request. The report shows 29 cast report lines were extracted and written to the data set associated with the LCALST DD name.

```
(C) 2002      Computer Associates International, Inc.      30OCT01 13:27:17  PAGE  1

                E N D E V O R   A P I   E X E C U T I O N
13:27:17  API0101I  BEGINNING OF API ACTION PROCESSING
          API0100I  LIST PACKAGE CAST REPORT
          API0100I  PACKAGE ID: PK1231
          API0100I  TO DDNAME: LCALST
          API0100I
13:27:17  API0102I  DISPATCHING API ACTION
13:27:18  API0000I  RETURNED COUNT=00029, SELECTED COUNT=00029
13:27:18  API0000I  PROCESSING COMPLETE - RC=00000 REASON=00000
```

The next example shows a list package correlation action request. The report shows 2 syntax errors were detected in the request block. A return code of 12 with a reason code of 2 is issued whenever syntax errors are detected.

```
(C) 2002      Computer Associates International, Inc.      28OCT01 15:55:11  PAGE  1

                E N D E V O R   A P I   E X E C U T I O N
15:55:11  API0101I  BEGINNING OF API ACTION PROCESSING
          API0100I  LIST PACKAGE CORRELATION
          API0100I  PACKAGE ID: PK1*
          API0100I  TO DDNAME: LCOLST
          API0100I  WHERE CORRELATION TYPE EQUAL: J
          API0100I
15:55:11  API0018E  THE ALCOR_RQ_PKGID FIELD VALUE IS BLANK OR CONTAINS A WILDCARD CHAR.
15:55:11  API0019E  THE ALCOR_RQ_COR_TYPE FIELD VALUE MUST BE H, I OR U. J IS INVALID.
15:55:11  API0000E  SYNTAX ERROR(S) DETECTED, PROCESSING TERMINATED
15:55:11  API0000I
15:55:11  API0000I  RETURNED COUNT=00000, SELECTED COUNT=00000
15:55:11  API0000I  PROCESSING COMPLETE - RC=00012 REASON=00002
```

3.2.5 Update Package Action Function Call Sample Reports

The following sample report formats are used to record the processing information for the API update package action function calls.

The reports consists of two parts: first, the data from the request structure is formatted and printed in a format similar to the output you receive when executing an Endeavor batch request; the second part is a summary of how many records were read and selected and the highest return and reason codes detected.

Below is a copy of a sample API execution report for an approve package action request. In this example, the action completed successfully. The return code and the reason code are set to zero.

```
(C) 2002      Computer Associates International, Inc.      28OCT01 12:28:51 PAGE  1

                E N D E V O R   A P I   E X E C U T I O N
12:28:51  API0101I  BEGINNING OF API ACTION PROCESSING
          API0100I  APPROVE PACKAGE
          API0100I  PACKAGE ID: PK1PKG87
          API0100I  MESSAGE DDNAME: PAPMSG
          API0100I
12:28:51  API0102I  DISPATCHING API ACTION
12:29:02  API0000I  PROCESSING COMPLETE - RC=00000 REASON=00000
```

The next example shoes a sample API execution report for a cast package action request. Notice 3 syntax errors were detected. A return code of 12 with a reason code of 2 is issued whenever syntax errors are detected.

```
(C) 2002      Computer Associates International, Inc.      01JUN01 9:08:31 PAGE  1

                E N D E V O R   A P I   E X E C U T I O N
09:08:32  API0101I  BEGINNING OF API ACTION PROCESSING
          API0100I  CAST PACKAGE
          API0100I  PACKAGE ID: PK1PKG06
          API0100I  MESSAGE DDNAME: PCAMSG
          API0100I  VALIDATE COMPONENTS: C
          API0100I  ENABLE BACKOUT: B
          API0100I  EXECUTION WINDOW FROM: 01JUN01 HH:MM TO: 30NOV01 23:59
          API0100I
09:08:32  API0003E  THE APCAS_RQ_VALCMP FIELD VALUE MUST BE Y, N OR W. C IS INVALID
09:08:32  API0003E  THE APCAS_RQ_BOENABLED FIELD VALUE MUST BE Y OR N. B IS INVALID
09:08:32  API0037E  THE FROM EXECUTION WINDOW DATE AND/OR TIME IS INVALID
09:08:32  API0000E  SYNTAX ERROR(S) DETECTED, PROCESSING TERMINATED
          API0100I
09:08:32  API0000I  PROCESSING COMPLETE - RC=00012 REASON=00002
```

The following report shows a create package correlation action request. In this example, the correlation record already existed. A return code of 12 with a reason code of 14 is issued whenever the condition is detected.

3.2 API Execution Reports

```
(C) 2002      Computer Associates International, Inc.      28OCT01 14:53:59  PAGE  1

                E N D E V O R   A P I   E X E C U T I O N
14:53:59  API0101I  BEGINNING OF API ACTION PROCESSING
          API0100I  CREATE CORRELATION
          API0100I  PACKAGE ID: PK1PKG54
          API0100I  MESSAGE DDNAME: PCOMSGC
          API0100I  CORRELATION TYPE: H
          API0100I  CORRELATION ID: PK1CORR54
          API0100I  CORRELATION DATA: ASSOCIATE ENDEVOR AND ALLFUSION HARVEST CM PACKAGE
          API0100I
14:53:59  API0102I  DISPATCHING API ACTION
14:54:00  PKMR020E  CORRELATION ALREADY EXISTS
14:54:00  API0000E  ERROR(S) DETECTED, PROCESSING TERMINATED
14:54:00  API0000I
14:54:00  API0000I  PROCESSING COMPLETE - RC=00012 REASON=00014
```

Below is a copy of a sample API execution report for a modify package correlation action request. In this example, the action completed successfully. The return code and the reason code are set to zero.

```
(C) 2002      Computer Associates International, Inc.      28OCT01 12:29:02  PAGE  1

                E N D E V O R   A P I   E X E C U T I O N
12:29:02  API0101I  BEGINNING OF API ACTION PROCESSING
          API0100I  MODIFY CORRELATION
          API0100I  PACKAGE ID: PK1PKG54
          API0100I  MESSAGE DDNAME: PCOMSGM
          API0100I  CORRELATION TYPE: H
          API0100I  CORRELATION ID: PK1CORR54
          API0100I  CORRELATION DATA: PREPARE TO EXECUTE BOTH PKGS ON 12/31/01
          API0100I
12:29:02  API0102I  DISPATCHING API ACTION
12:29:02  API0000I  PROCESSING COMPLETE - RC=00000 REASON=00000
```

The next example shows an execute package request. In this report, the action completed successfully. The return code and the reason code are set to zero. A second response file, C1EXMSGs, is also created. It contains the standard Endeavor Syntax Request Report, the Endeavor Execution Report and the Endeavor Action Summary Report that are produced when a package is executed in foreground or background mode.

```
(C) 2002      Computer Associates International, Inc.      30OCT01 15:35:13  PAGE  1

                E N D E V O R   A P I   E X E C U T I O N
15:35:13  API0101I  BEGINNING OF API ACTION PROCESSING
          API0100I  EXECUTE PACKAGE
          API0100I  PACKAGE ID: PK1PKG54
          API0100I  MESSAGE DDNAME: APIMSGS
          API0100I
15:35:13  API0102I  DISPATCHING API ACTION
15:35:27  API0000I  PROCESSING COMPLETE - RC=00000 REASON=00000
```

3.3 The API Diagnostic Trace — BC1PAPI

The API Diagnostic Trace records diagnostic and informational messages for each API transaction. The trace should be run only at the request of Endeavor technical support. To activate the Endeavor API Diagnostic Trace, include the following DD statement in your JCL stream:

```
//BSTAPI DD SYSOUT=A
```

Below is an example of a response file created with the BC1PAPI trace facility.

```
(C) 2002 Computer Associates International Endeavor 02/04/01 07:34:00 PAGE 1
DIAGNOSTICS OUTPUT LISTING
BC1PAPI - INITIALIZING - VERSION 01/21/99 12.49
BC1PAPI SEND MESSAGE REQUEST
BC1PAPI APISTG $BGETSTG RC=00004 RESULT=NEWSTG
BC1PAPI MAIN2000 $PINQ RC=00012
BC1PAPI MAIN2000 $PINIT RC=00000
BC1PAPI MAIN2100 $PGET RC=00000
BC1PAPI BMSG2000 - 0040 0000 000A 0001 $CTL 0006EF20
BC1PAPI BMSG2000 - 0050 0000 0450 0001 LELQ 0006EF60
BC1PAPI BMSG2000 - 039C 0000 0451 0001 LELR 8006EFB0
BC1PAPI MAIN2300 $SEND RC=00004
BC1PAPI ISSUING ATTACH FOR API SERVER
BC1PAPI ATTA1000 $PWAIT RC=00000
BC1PAPI ATTA2000 $PRECV RC=00000
BC1PAPI ATTA3000 $PFREE RC=00000
BC1PAPI MAIN2300 $SEND RC=00000
```

3.4 The API Internal Trace — EN\$TRAPI

The API Internal Trace records detailed internal trace information along with API data block dumps. This trace should only be used to debug a problem.

To activate the Endeavor API Internal Trace, include the following DD statement in your JCL stream:

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

Below are examples of response files created with the EN\$TRAPI trace facility.

```
(C) 2002      Computer Associates International      Endeavor      02/04/01  07:34:04      PAGE      1

                                     ENDEVOR API INTERNAL TRACE
ENAPIMGR - INITIALIZING - VERSION 01/21/99 12.41
ENAPIMGR INIT3000 $PINIT RC=00000
ENAPIMGR INIT4000 $PHDL RC=00000
ENAPIMGR INIT5000 $PGET RC=00000
ENAPIMGR INIT6000 $SEND RC=00000
ENAPIMGR INIT7000 $MSGIORTN INTERCEPT ESTABLISHED
ENAPIMGR PROC1000 $PRECV RC=00000
ENAPIMGR PROC3000 COMPLETE MESSAGE RECEIVED - ADDRESS=00049154 LENGTH=045C
00049154 (+0000) 0060045C 5BD4E2C7 00000003 00000000 00000000 00000000 00000000 00000000 * - *$MSG *
00049174 (+0020) 00000000 00000000 00000000 00000000 00400000 000A0001 5BC3E3D3 D4E2C7F3 * $CTLMSG3 *
00049194 (+0040) C6C9D3C5 C5E7E3F1 C5D3D440 40404040 40404040 40404040 F0F0F0F0 F0F0F0F0 *FILEEXTIELM 00000000*
000491B4 (+0060) F0F0F0F0 F0F0F0F0 F0F0F0F0 F0F0F0F0 00500000 04500001 D3C5D3D8 D7C1C140 *000000000000000000 & & LELQPAA *
000491D4 (+0080) C9D5E340 40404040 D5C4E5D9 D4E5E240 C2C1E2C5 40404040 C1E2D4D7 C7D44040 *INT NDVRMVS BASE ASMPGM *
000491F4 (+00A0) C2C3F1D7 C9D4C7D9 4040F140 40404040 40404040 C1C2C3C4 C6C74040 40404040 *BC1PIMGR 1 ABCDFG *
00049214 (+00C0) 039C0000 04510001 D3C5D3D9 40404040 40404040 40404040 40404040 * LELR *
```

The following example contains additional trace information for an ADD element action function call.

```
(C) 2002      Computer Associates International      Endeavor      02/04/01  07:34:04      PAGE      1

                                     ENDEVOR API INTERNAL TRACE
ENAPIMGR CALL1020 MSG PRB, ADDMSG , IS SUCCESSFULLY SETUP
ENAPIACT XADDRUTN -- ADD ACTION ROUTINE
ENAPIACT PRINTRTN - PRINT ACTION REQUEST BLOCK ROUTINE
ENAPIACT PRINTRTN - CALLING ENAPIPRA MODULE
ENAPIACT PRINTRTN - CALL TO ENAPIPRA COMPLETE. RC=00000
ENAPIACT PRINTRTN COMPLETE - RC=00000 RE=00000
ENAPIACT DACTRUTN - DISPATCH ACTION ROUTINE
ENAPIACT DACTRUTN - CALLING C1B4100 MODULE
ENAPIACT DACTRUTN - CALL TO C1B4100 COMPLETE. RC=00000
ENAPIACT DACTRUTN - CALLING C1B4200 MODULE
ENAPIACT DACTRUTN - ERRORS DETECTED BY CALLED MODULE
ENAPIACT DACTRUTN COMPLETE - RC=00004 RE=00000
ENAPIACT XADDRUTN COMPLETE - RC=00004 RE=00000
ENAPIMGR CALL4000 ROUTINE RC=00004 RE=00000
ENAPIMGR PROC4000 COMPLETE MESSAGE SENT - ADDRESS=0004B154
```

Appendix A. Sample API Programs

A.1 Overview

This appendix provides a description of sample programs and JCL that you can use to test the Endeavor API. The appendix includes:

- A description of how to execute an API program outside of a processor using the NDVRC1 program.
- A description of COBOL program CCIDRPT1 that produces a list of elements and creates a CCID cross-reference report. The JCL to execute program CCIDRPT1.
- A description of Assembler program ENHAAPGM that produces a list of environments and writes the responses to a response file. The JCL to execute program ENHAAPGM.
- A description of Assembler program ENHAEPGM that executes each of the element action function calls and writes the responses to a response file. The JCL to execute program ENHAEPGM.
- A description of Assembler program ENHAPLST that executes the list package action function calls and writes the responses to a response file. The JCL to execute program ENHAPLST.
- A description of Assembler program ENHAPUPD that executes the update package action function calls and writes the responses to a response file. The JCL to execute program ENHAPUPD.
- A description of Assembler program ENTBJAPI that executes different inventory list function calls and writes the responses to an output file. The JCL to execute program ENTBJAPI.

A.2 Executing an API Program

The NDVRC1 program allows you to execute a program that issues Endeavor API function calls outside of a processor.

A.2.1 Description

You must execute NDVRC1 and pass the name of your program through the PARM= parameter on the EXEC statement. If your program requires parameter data, you can append it to the parameter string using a comma to separate the program name from your parameter data.

For example:

```
//STEP1 EXEC PGM=NDVRC1,PARM='TESTAPI1,DATA1,DATA2'
```

NDVRC1 executes program TESTAPI1 and passes the following parameter information to the program through register 1:

R1 = 00081010

at address 81010 the following is found: 00081020

at address 81020 the following data (shown in HEX) can be found:

000C6BC4C1E3C1F16BC4C1E3C1F2

where:

length value = 12

parm data = ,data1,data2

NDVRC1 reserves the first eight characters of the PARM parameter for the program name. All other parameters, starting with the ninth character, are passed to the API program as parameter data. For Assembler programs, register 1 contains an address that points to the parameter data where the first two bytes contain the parameter length followed by the parameter data.

Note: If you do not want the first comma passed to the program, enter the data immediately after the program name and omit entering the first comma.

COBOL Users: NDVRC1 reserves the first eight characters of the PARM parameter for the program name. All other parameters, starting with the ninth character, are passed to the API program and placed into the LINKAGE SECTION storage provided in the API program. The API program must contain the PROCEDURE DIVISION USING storage clause, where storage is the 01-level name of the variable specified in the LINKAGE SECTION.

Using the same parameter information as shown in the Assembler example, the storage defined in the LINKAGE SECTION must be defined as 14 bytes in length and contain

the two byte-length value (000C) in binary, PIC 9(2) COMP, followed by the 12 bytes of parameter data (,data1,data2) in character format, PIC X(12).

A.3 Sample COBOL Program — CCIDRPT1

The COBOL program, CCIDRPT1, produces a list of elements based on user input and creates a CCID cross-reference report. This program was written to show an application use of the API feature. JCL that you can use to execute this program appears in A.3.3, “JCL to Execute CCIDRPT1 — BC1JRAPI” on page A-7.

A.3.1 Description

The source for this program is distributed with Endeavor as member name CCIDRPT1 in the iprfx.iqual.SOURCE data set.

The CCIDRPT1 program performs the following actions:

1. Issues an API list element function call and writes the responses to a response file.
2. Reads each response record.
3. Issues an API extract element function call specifying the change option. A second file is created containing the changes associated with each element.
4. Extracts the CCID data and writes it to a sort file; the CCID data precedes the element source statements.
5. Reads the sort file and generates a report.

Only source code is provided for this program. Review the source and make any desired modifications. You must compile and link-edit this module into the uprfx.uqual.AUTHLIB library before attempting to execute it. There are no restrictions on linkage editor AMODE/RMODE parameters. (AMODE=31,RMODE=ANY or AMODE=24,RMODE=24).

A.3.2 CCIDRPT1 Output Report

Below is a portion of the report output from program CCIDRPT1:

```

                                ENDEVOR INVENTORY REPORT BY CCID
                                PAGE 1

*****
*****
*****
*
* THIS REPORT PRODUCES A CCID CROSS REFERENCE ON ELEMENT
* INVENTORY SPECIFIED BY THE USER. SEE SPECIFICATION INPUT
* BELOW.
*
*****
*****
*****

USER SPECIFICATIONS:
  FROM ENVIRON:QAS      STGID:2 SYSTEM:NDVRMVS  SUBSYS:BASE  ELEMENT:*  TYPE:ASMPGM
  TO ENVIRON:          STGID:
  SEARCH SETTING - CURRENT LOCATION
  PATH SETTING - PHYSICAL
  PROCESS FIRST OCCURRENCES OF ELEMENT AND TYPE
    
```

```

                                ENDEVOR INVENTORY REPORT BY CCID
                                PAGE 2

CCID: I5797110

ELEMENT  TYPE  VV.LL DATE  TIME  USER  ENVIRON  SID  SYSTEM  SUBSYS  COMMENT
C1GP2000 ASMPGM  01.54 24OCT99 13:33 OLEJU01 QAS    2  NDVRMVS  BASE   disallow batch adm/ pkg w/in processors

CCID: NMAN

ELEMENT  TYPE  VV.LL DATE  TIME  USER  ENVIRON  SID  SYSTEM  SUBSYS  COMMENT
C1GSCIP0 ASMPGM  01.17 24JUL99 13:15 OLEJU01 QAS    2  NDVRMVS  BASE   ADD LOGIC FOR NAME CCID VALID FLAG

CCID: OFT

ELEMENT  TYPE  VV.LL DATE  TIME  USER  ENVIRON  SID  SYSTEM  SUBSYS  COMMENT
ENCOPTBL ASMPGM  01.09 16OCT99 14:25 BUCFR02 QAS    2  NDVRMVS  BASE   Optional Features Table Source

CCID: PKGESI

ELEMENT  TYPE  VV.LL DATE  TIME  USER  ENVIRON  SID  SYSTEM  SUBSYS  COMMENT
CONMSGSP ASMPGM  01.48 05SEP96 11:44 DYSR001 QAS    2  NDVRMVS  BASE   Add package ESI support
C1SPMISC ASMPGM  01.38 31OCT99 10:33 BUCFR02 QAS    2  NDVRMVS  BASE   ESI SUPPORT
ENMP3CRE ASMPGM  01.03 21JUL99 09:11 DYSR001 QAS    2  NDVRMVS  BASE   Package exit 1 support

CCID: P0000644

ELEMENT  TYPE  VV.LL DATE  TIME  USER  ENVIRON  SID  SYSTEM  SUBSYS  COMMENT
C1BML010 ASMPGM  01.28 20MAY96 09:09 BERBE02 QAS    2  NDVRMVS  BASE   S0c4 at +82 during clear footprint cmd.

CCID: P0000701

ELEMENT  TYPE  VV.LL DATE  TIME  USER  ENVIRON  SID  SYSTEM  SUBSYS  COMMENT
C1BR2000 ASMPGM  01.25 30DEC99 11:00 BERBE02 QAS    2  NDVRMVS  BASE   zero days value for reports

CCID: P0000894

ELEMENT  TYPE  VV.LL DATE  TIME  USER  ENVIRON  SID  SYSTEM  SUBSYS  COMMENT
ENBX1400 ASMPGM  01.19 24DEC99 16:26 BERBE02 QAS    2  NDVRMVS  BASE   included copybooks have space b4 period
    
```

A.3.3 JCL to Execute CCIDRPT1 — BC1JRAPI

The JCL for this program is distributed with Endeavor as member name BC1JRAPI in the iprfx.igual.JCL data set. This job shows how to execute program CCIDRPT1. The load module along with its source is distributed.

The JCL to execute program CCIDRPT1 appears below. Look at the JCL comment statements that describe updates you need to make before you execute this JCL stream.

In this example, DD name, MSG3FILE, describes the message response file and DD name, EXT1ELM, describes the response file for the element extract responses.

```

//* ( COPY JOBCARD )
/******
/*
/*      BC1JRAPI - THIS IS SAMPLE JCL THAT EXECUTES PROGRAM CCIDRPT1
/*              WHICH IS DISTRIBUTED IN SOURCE AS AN EXAMPLE OF
/*              HOW TO WRITE (USE) THE ENDEVOR API FACILITY.
/*
/*      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 STEPLIB AND CONLIB DATA SET NAMES ARE
/*         CORRECT.
/*         - uprfx.igual.AUTHLIB
/*         - iprfx.igual.AUTHEXT
/*         - iprfx.igual.CONLIB
/*      3. MODIFY THE PROGRAMS DATA INPUT. AS AN EXAMPLE, THE
/*         INPUT DATA HAS A SINGLE REQUEST.
/******
//STEP1   EXEC PGM=NDVRC1,PARM='CCIDRPT1',DYNAMNBR=1500,REGION=4096K
//STEPLIB DD DSN=uprfx.igual.AUTHLIB,DISP=SHR
//        DD DSN=iprfx.igual.AUTHEXT,DISP=SHR
//CONLIB  DD DSN=iprfx.igual.CONLIB,DISP=SHR
//SYSOUT  DD SYSOUT=*

```

```

//SYSPRINT DD  SYSOUT=*
//BSTERR  DD  SYSOUT=*
//MSG3FILE DD  DSN=&&MSG3FILE,DISP=(NEW,PASS),
//          UNIT=SYSDA,SPACE=(TRK,(5,5)),
//          DCB=(RECFM=FB,LRECL=133,BLKSIZE=13300)
//EXT1ELM DD  DSN=&&EXT1ELM,DISP=(NEW,PASS),
//          UNIT=SYSDA,SPACE=(TRK,(5,5)),
//          DCB=(RECFM=VB,LRECL=2048,BLKSIZE=22800)
//EXT2ELM DD  DSN=&&EXT2ELM,DISP=(NEW,PASS),
//          UNIT=SYSDA,SPACE=(TRK,(5,5)),
//          DCB=(RECFM=VB,LRECL=2048,BLKSIZE=22800)
//WORKELM DD  DSN=&&WORKELM,DISP=(NEW,PASS),
//          UNIT=SYSDA,SPACE=(TRK,(5,5)),
//          DCB=(RECFM=FB,LRECL=115,BLKSIZE=1150)
//SORTELM  DD  DSN=&&SORTELM,DISP=(NEW,PASS),
//          UNIT=SYSDA,SPACE=(TRK,(5,5)),
//          DCB=(RECFM=FB,LRECL=115,BLKSIZE=1150)
//CCIDRPT1 DD  SYSOUT=*
//*
/** SYSIN DATA STRUCTURE INFORMATION
/**
/** COLUMN 1      = '*'      STATEMENT TREATED AS A LINE COMMENT & WILL
/**                                     ON THE REPORT
/**
/** COLUMNS 1-4  = 'ELOC'   INPUT REQUEST ID
/** COLUMN 5      = PATH SETTING
/**               'L' = LOGICAL PATH
/**               'P' = PHYSICAL PATH
/** COLUMN 6      = SEARCH SETTING
/**               'A' = SEARCH ALL
/**               'B' = SEARCH BETWEEN
/**               'N' = NO SEARCH
/**               'E' = SEARCH NEXT
/**               'R' = SEARCH RANGE
/** COLUMN 7      = RETURN SETTING
/**               'A' = RETURN ALL HITS
/**               'F' = RETURN FIRST HIT
/** COLUMNS 8-15 = ENVIRONMENT NAME (EXPLICIT)
/** COLUMN 16     = STAGE ID (EXPLICIT)
/** COLUMNS 17-24 = SYSTEM NAME (MAY BE WILD)
/** COLUMNS 25-32 = SUBSYSTEM NAME (MAY BE WILD)
/** COLUMNS 33-42 = ELEMENT NAME (MAY BE WILD)
/** COLUMNS 43-50 = TYPE NAME (MAY BE WILD)
/** COLUMNS 51-58 = TO ENVIRON NAME SPECIFY ON SEARCH ='B' OR 'R'
/**                                     & MUST BE EXPLICIT
/** COLUMN 59     = TO STAGE ID SPECIFY ON SEARCH ='B' OR 'R'
/**                                     & MUST BE EXPLICIT
/** COLUMNS 60-69 = TO ELEMENT NAME (MAY BE WILD)

```

```
//* +---1---+---2---+---3---+---4---+---5---+---6---+---7---
//SYSIN DD *
*****
*****
*****
*
* THIS REPORT PRODUCES A CCID CROSS REFERENCE ON ELEMENT *
* INVENTORY SPECIFIED BY THE USER, SEE SPECIFICATION INPUT BELOW. *
*
*****
*****
*****
ELOCPNFINT INDVRMVS BASE * ASMPGM
//* PRINT ANY MESSAGES
//STEP2 EXEC PGM=IEBGENER
//SYSPRINT DD DUMMY
//SYSIN DD DUMMY
//SYSUT1 DD DSN=&&MSG3FILE,DISP=(OLD,DELETE)
//SYSUT2 DD SYSOUT=*
```

A.4 Sample List Environment Function Call — ENHAAPGM

The sample Assembler program, ENHAAPGM, produces a list of environments and writes the responses to a response file. You can use this program as a template for creating other inventory list function calls. For example, you can modify the request structure field values to produce a list of types or add an additional function call to produce a list of systems. JCL to execute this program appears in A.4.2, “JCL to Execute ENHAAPGM — BC1JAPGM”

A.4.1 Description

The source for this program is distributed with Endeavor as member name ENHAAPGM in the iprfx.igual.SOURCE data set. The ENHAAPGM program performs the following actions:

1. Issues a request to the API to read the MCF and build a list of all the environments.
2. Writes the list of environments to a response file.
3. Writes any inventory or source management messages to the message data set.

Only source code is provided for this program. Review the source and make any necessary modifications. The names of the starting and ending environments and the search options are candidates for modifications. You must assemble and link-edit this module into uprfx.igual.AUTHLIB before attempting to execute it. There are no restrictions on linkage editor AMODE/RMODE parameters (AMODE=31,RMODE=ANY or AMODE=24,RMODE=24).

A.4.2 JCL to Execute ENHAAPGM — BC1JAPGM

The JCL in this section is distributed with Endeavor as member name BC1JAPGM in the iprfx.igual.JCL data set. This job can be tailored and used to execute the ENHAAPGM program. The sample API program must be executed from an authorized library.

The JCL to execute program ENHAAPGM appears below. In this example, the DD name APIMSGS describes the message file, in this case SYSOUT, and APILIST describes the response file for the list environment function call.

```

/**(JOB CARD)
/**-----*
/**
/** (C) 2002      COMPUTER ASSOCIATES INTERNATIONAL, INC.
/**
/** NAME: BC1JAPGM
/**
/** PURPOSE - THIS IS SAMPLE JCL TO INVOKE THE ASSEMBLER
/**          SAMPLE API PROGRAM: ENHAAPGM
/**
/** THE FOLLOWING UPDATES MUST BE MADE TO THIS JCL BEFORE
/** IT CAN BE EXECUTED:
/**
/** 1. UPDATE THE JOB CARD TO REFLECT CORRECT SITE INFORMATION
/** 2. REVIEW THAT THE STEPLIB DD STATEMENT CONCATENATION CONTAINS
/**    THE NAME OF THE AUTHORIZED LIBRARY CONTAINING THE SAMPLE
/**    PROGRAM AND THE ENDEVOR DEFAULTS TABLE.
/** 3. REVIEW THAT THE CONLIB DD STATEMENT CONCATENATION CONTAINS
/**    THE NAME OF THE LIBRARY CONTAINING THE ENDEVOR SOFTWARE.
/**-----*
//STEP1 EXEC PGM=ENHAAPGM,REGION=4096K
//STEPLIB DD DISP=SHR,DSN=uprfx.uqua1.AUTHLIB
//          DD DISP=SHR,DSN=iprfx.iqua1.AUTHLIB
//CONLIB DD DISP=SHR,DSN=iprfx.iqua1.CONLIB
//SYSOUT DD SYSOUT=*
//SYSPRINT DD SYSOUT=*
//BSTERR DD SYSOUT=*
//APIMSG DD SYSOUT=*,
//          DCB=(RECFM=FB,LRECL=133,BLKSIZE=13300)
//APIEXTR DD DSN=&&EXT1ELM,DISP=(NEW,PASS),
//          UNIT=SYSDA,SPACE=(TRK,(5,5)),
//          DCB=(RECFM=VB,LRECL=2048,BLKSIZE=22800)
/**          TRACE FACILITY
/**-----*
//*BSTAPI DD SYSOUT=*
//*EN$TRAPI DD SYSOUT=*

/**-----*
/** PRINT EXTRACTED LIST OF ENVIRONMENTS
/**-----*
//STEP2 EXEC PGM=IEBGENER
//SYSPRINT DD DUMMY
//SYSIN DD DUMMY
//SYSUT1 DD DSN=&&EXT1ELM,DISP=(OLD,DELETE)
//SYSUT2 DD SYSOUT=*

```

A.5 Sample Element Action Function Call — ENHAEPGM

The sample Assembler program, ENHAEPGM, illustrates the use of all the API element action function calls. JCL to execute this program appears in A.5.2, “JCL to Execute ENHAEPGM — BC1JEPGM”

A.5.1 Description

The source for this program is distributed with Endeavor as member name ENHAEPGM in the iprfx.iqual.SOURCE data set. The ENHAEPGM program contains logic to execute each of the element action function calls supported by the API.

Only source code is provided for this program. Review the source and make any necessary modifications. In order for this program to operate, the name of the input and output data sets must be modified and the name of the member being ADDED and UPDATED must exist in the input library. Also, the inventory location information (ENV, SYS, SYSSUB, etc.) must exist.

You must assemble and link-edit this module into uprfx.uqual.AUTHLIB before attempting to execute it. There are no restrictions on the linkage editor AMODE/RMODE parameters. AMODE=31,RMODE=ANY is recommended.

A.5.2 JCL to Execute ENHAEPGM — BC1JEPGM

The JCL in this section is distributed with Endeavor as member name BC1JEPGM in the iprfx.iqual.JCL data set. This job can be tailored and used to execute the ENHAEPGM program. This sample API program must be executed from an authorized library.

The JCL to execute program ENHAEPGM appears below. In this example, a series of DD names describes the message, input and response files defined in the sample program. Also, the optional trace DD statements are coded.

```

/*(JOB CARD)
/*-----*
/*
/* (C) 2002    COMPUTER ASSOCIATES INTERNATIONAL, INC.
/*
/* NAME: BC1JEPGM
/*
/* PURPOSE - THIS IS SAMPLE JCL TO INVOKE THE ASSEMBLER
/*          SAMPLE API ELEMENT ACTION PROGRAM: ENHAEPGM
/*
/* THE FOLLOWING UPDATES MUST BE MADE TO THIS JCL BEFORE
/* IT CAN BE EXECUTED:
/*
/* 1. UPDATE THE JOB CARD TO REFLECT CORRECT SITE INFORMATION
/* 2. REVIEW THAT THE STEPLIB DD STATEMENT CONCATENATION CONTAINS
/*    THE NAME OF THE AUTHORIZED LIBRARY CONTAINING THE SAMPLE
/*    PROGRAM AND THE ENDEVOR DEFAULTS TABLE.
/* 3. REVIEW THAT THE CONLIB DD STATEMENT CONCATENATION CONTAINS
/*    THE NAME OF THE LIBRARY CONTAINING THE ENDEVOR SOFTWARE.
/* 4. THE SAMPLE PROGRAM EXPECTS SEVERAL MEMBERS TO EXIST IN THE
/*    UPRFX.UQUAL.SRCLIB. MAKE SURE THIS LIBRARY EXISTS AT YOUR
/*    SITE OR CHANGE THE DATA SET NAME IN THIS JCL TO ONE THAT
/*    DOES AND CREATE 3 COPYBOOK MEMBERS IN THIS LIBRARY;
/*      APIB--BASE COPYBOOK
/*      APIU--BASE COPYBOOK WITH SEVERAL MODIFICATIONS
/*      APIR--COPY OF APIB FOR DELETE AND RETRIEVE ACTIONS
/*-----*
//STEP1 EXEC PGM=NDVRC1,PARM='ENHAEPGM',REGION=4096K
//STEPLIB DD DISP=SHR,DSN=UPRFX.UQUAL.AUTHLIB (C1DEFLT, USER PGM)
//        DD DISP=SHR,DSN=IPRFX.IQUAL.AUTHLIB (AUTH PGMS)
//CONLIB DD DISP=SHR,DSN=IPRFX.IQUAL.LOADLIB (SOFTWARE)

```

```

//*****
//*      API MESSAGES DD STMT FROM THE AACTL MSG_DD CONTROL FIELD.
//*      IF NOT SPECIFIED, DD NAME DEFAULT OF APIMSGS IS USED.
//*      EACH REQUEST SHOULD HAVE A UNIQUE MESSAGE FILE.
//*****
//APIMSGS DD SYSOUT=*,DCB=(RECFM=FBA,LRECL=133,BLKSIZE=0)
//ADDMSG  DD SYSOUT=*
//DELMMSG DD SYSOUT=*
//GENMSG  DD SYSOUT=*
//MOVMSG  DD SYSOUT=*
//PREMSG  DD SYSOUT=*
//PRMMSG  DD SYSOUT=*
//RETMSG  DD SYSOUT=*
//SIGMSG  DD SYSOUT=*
//TRAMSG  DD SYSOUT=*
//UPDMSG  DD SYSOUT=*
//*****
//*      TO/FROM DD NAMES SPECIFIED ON THE ELEMENT ACTION REQUESTS
//*****
//ADDDNMI DD DISP=SHR,DSN=UPRFX.UQUAL.SRCLIB
//PREDDNMO DD SYSOUT=*,DCB=(RECFM=FBA,LRECL=133,BLKSIZE=0)
//PRMDDNMI DD DISP=SHR,DSN=UPRFX.UQUAL.SRCLIB
//PRMDDNMO DD SYSOUT=*
//RETDDNMO DD DISP=SHR,DSN=UPRFX.UQUAL.SRCLIB
//UPDDNMI DD DISP=SHR,DSN=UPRFX.UQUAL.SRCLIB
//*****
//*      OPTIONAL TRACE DD STATEMENTS
//*****
//BSTERR  DD SYSOUT=*          (ENDEVOR TRACE)
//BSTAPI  DD SYSOUT=*          (API DIAGNOSTIC TRACE)
//EN$TRAPI DD SYSOUT=*          (API INTERNAL TRACE)

```

A.6 Sample List Package Action Function Call — ENHAPLST

This Assembler program, ENHAPLST, illustrates the use of all the list package action functions of the API. JCL to execute this program appears in A.6.2, “JCL to Execute ENHAPLST — BC1JPLST.”

A.6.1 Description

This program operates at your site with minor modifications. As delivered, a package id of PKGIDVALUE is specified for each of the package id request fields. You need to modify these values to a package id that exists at your site. The request for the package header information specifies a package id value of PKGID* to illustrate the use of wildcarding. We recommend you modify this value to be a partially wildcarded value that causes a reasonable number of packages to be selected.

A.6.2 JCL to Execute ENHAPLST — BC1JPLST

The JCL in this section is distributed with Endeavor as member name BC1JPLST in the iprfx.igual.JCLLIB data set. This job shows how to execute program ENHAPLST.

In this example, NDVRC1 is executed and passes a parameter containing the name of the API test program, ENHAPLST. You can also invoke Endeavor API applications this way. This API program uses a different message and response file for each request (*aaaMSG* and *aaaLST*, where *aaa* is the function such as LPK, LCA, and LSC). The trace DD statements, BSTAPI and EN\$TRAPI also exist, but are commented out.

```

//*JOB CARD GOES HERE
/*-----*
/*
/* (C) 2002      COMPUTER ASSOCIATES INTERNATIONAL, INC.      *
/*
/* NAME: BC1JPLST                                           *
/*
/* PURPOSE - THIS IS SAMPLE JCL TO INVOKE THE ASSEMBLER      *
/*           SAMPLE API PACKAGE ACTION PROGRAM: ENHAPLST      *
/*           THE PROGRAM NAME AND THE REQUIRED DD STATEMENTS WILL*
/*           VARY SINCE THIS IS A USER WRITTEN PROGRAM.      *
/*
/* 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 STEPLIB DD STATEMENT CONCATENATION CONTAINS *
/*    THE NAME OF THE AUTHORIZED LIBRARY CONTAINING THE SAMPLE *
/*    PROGRAM AND THE ENDEVOR DEFAULTS TABLE.                *
/* 3. REVIEW THAT THE OPTIONAL LIST DATA SET NAMES AND THEIR *
/*    ATTRIBUTES ARE CORRECT.                                  *
/*-----*

```

```
//STEP1 EXEC PGM=NDVRC1,PARM='ENHAPLST',REGION=4096K
//STEPLIB DD DISP=SHR,DSN=UPRFX.UQUAL.AUTHLIB (C1DEFLTS,USER PGMS)
// DD DISP=SHR,DSN=IPRFX.IQUAL.AUTHLIB (AUTH PGMS)
//CONLIB DD DISP=SHR,DSN=IPRFX.IQUAL.CONLIB (SOFTWARE)
//* API MESSAGES DD STMT FROM THE USER CONTROL BLOCK
//*APIMSGS DD SYSOUT=*,DCB=(RECFM=FBA,LRECL=133,BLKSIZE=0)
//*APILST DD DISP=(,CATLG,DELETE),DSN=UPRFX.UQUAL.APIMSGS,
//* UNIT=PDISK,SPACE=(TRK,(2,1),RLSE),
//* DCB=(RECFM=VB,LRECL=2048,BLKSIZE=0),DSORG=PS
//LAPMSG DD SYSOUT=*
//LAPLST DD DISP=SHR,DSN=UPRFX.UQUAL.APILAPP
//LBKMSG DD SYSOUT=*
//LBKLIST DD DISP=SHR,DSN=UPRFX.UQUAL.APILBKO
//LCAMSG DD SYSOUT=*
//LCALST DD DISP=SHR,DSN=UPRFX.UQUAL.APILCAS
//LCOMSG DD SYSOUT=*
//LCOLST DD DISP=SHR,DSN=UPRFX.UQUAL.APILCOR
//LSCMSG DD SYSOUT=*
//LSCLST DD DISP=SHR,DSN=UPRFX.UQUAL.APILSCL
//LSUMSG DD SYSOUT=*
//LSULST DD DISP=SHR,DSN=UPRFX.UQUAL.APILSUM
//* ERROR AND TRACE DD STATEMENTS
//BSTERR DD SYSOUT=* (ENDEAVOR ERROR LOG)
//*EN$TRAPI DD SYSOUT=* (API INTERNAL TRACE)
//*BSTAPI DD SYSOUT=* (API DIAGNOSTIC TRACE)
```

A.7 Sample Update Package Action Function Call — ENHAPUPD

This Assembler program, ENHAPUPD, illustrates the use of all the update type package action functions of the API. This includes approve a package, backout and backin a package, commit a package, delete a package, deny a package, create, modify and delete a correlation, execute a package and reset a package. JCL to execute this program appears in A.7.2, “JCL to Execute ENHAPUPD — BC1JPUPD.”

A.7.1 Description

This program operates at your site with minor modifications. As delivered, a package id of PKGIDVALUE is specified for each of the package id request fields. You need to modify these values to a package id that exists at your site.

A.7.2 JCL to Execute ENHAPUPD — BC1JPUPD

The JCL in this section is distributed with Endeavor as member name BC1JPUPD in the iprfx.iqual.JCLLIB data set. This job shows how to execute program ENHAPUPD.

In this example, NDVRC1 is executed and passes a parameter containing the name of the API test program, ENHAPUPD. You can also invoke Endeavor API applications this way. This API program uses a different message file for each request (*aaa*MSG, where *aaa* is the function PAP, PBI, PBO, PCA, PCM, PCO, PDE, PDN, PEX and PRE). The C1EXMSGS DD statement is required to log the Endeavor messages created by the execute package action. The trace DD statements, BSTAPI and EN\$TRAPI also exist, but are commented out.

```

//*JOB CARD GOES HERE
/*-----*
/*
/* (C) 2002      COMPUTER ASSOCIATES INTERNATIONAL, INC.      *
/*
/* NAME: BC1JPUPD                                           *
/*
/* PURPOSE - THIS IS SAMPLE JCL TO INVOKE THE ASSEMBLER      *
/*           SAMPLE API PACKAGE ACTION PROGRAM: ENHAPUPD.    *
/*           THE PROGRAM NAME AND THE REQUIRED DD STATEMENTS WILL*
/*           VARY SINCE THIS IS A USER WRITTEN PROGRAM.      *
/*
/* THE FOLLOWING UPDATES MUST BE MADE TO THIS JCL BEFORE     *
/* IT CAN BE EXECUTED:                                       *
/*
/* 1. UPDATE THE JOB CARD TO REFLECT CORRECT SITE INFORMATION *
/* 2. REVIEW THAT THE STEPLIB DD STATEMENT CONCATENATION CONTAINS *
/*    THE NAME OF THE AUTHORIZED LIBRARY CONTAINING THE SAMPLE *
/*    PROGRAM AND THE ENDEVOR DEFAULTS TABLE.                *
/* 3. REVIEW THAT THE CONLIB DD STATEMENT CONCATENATION CONTAINS *
/*    THE NAME OF THE LIBRARY CONTAINING THE ENDEVOR SOFTWARE. *

```

```
/*-----*
//STEP1 EXEC PGM=NDVRC1,PARM='ENHAPUPD',REGION=4096K
//STEPLIB DD DISP=SHR,DSN=UPRFX.UQUAL.AUTHLIB (C1DEFLTS,USER PGMS)
// DD DISP=SHR,DSN=IPRFX.IQUAL.AUTHLIB (AUTH PGMS)
//CONLIB DD DISP=SHR,DSN=IPRFX.IQUAL.CONLIB (SOFTWARE)
//* API MESSAGES DD STMT FROM THE USER CONTROL BLOCK
//*APIMSGS DD SYSOUT=*,DCB=(RECFM=FBA,LRECL=133,BLKSIZE=0)
//PAPMSG DD SYSOUT=* (APPROVE)
//PBOMSG DD SYSOUT=* (BACKOUT)
//PBIMSG DD SYSOUT=* (BACKIN)
//PCAMSG DD SYSOUT=* (CAST)
//PCMMSG DD SYSOUT=* (COMMIT)
//PCOMSGC DD SYSOUT=* (CREATE CORRELATION)
//PCOMSGD DD SYSOUT=* (DELETE CORRELATION)
//PCOMSGM DD SYSOUT=* (MODIFY CORRELATION)
//PDEMSG DD SYSOUT=* (DELETE)
//PDNMSG DD SYSOUT=* (DENY)
//PEXMSG DD SYSOUT=* (EXECUTE)
//PREMSG DD SYSOUT=* (RESET)
//* ENDEVOR EXECUTE PACKAGE ACTION MESSAGES DD STMT
//C1EXMSG DD SYSOUT=*
//* ERROR AND TRACE DD STATEMENTS
//BSTERR DD SYSOUT=* (ENDEVOR ERROR LOG)
//*EN$TRAPI DD SYSOUT=* (API INTERNAL TRACE)
//*BSTAPI DD SYSOUT=* (API DIAGNOSTIC TRACE)
```

A.8 Sample Inventory List Function Call — ENTBJAPI

This program allows you to specify input statements to execute inventory list function calls. Based on user input, it builds control, request, and response structures and executes the API to process the function calls. It writes the responses to a response file. JCL to execute this program appears in A.8.2, “JCL to Execute ENTBJAPI — BC1JAAPI.”

A.8.1 Description

See comment statements in the JCL stream for a full description of each available API inventory list function call. Each function call is made up of a control structure statement, a request structure statement, and a RUN statement. The API structures are created from the control structure statement and the request structure statement. The RUN statement executes the API. More than one function call can be executed in one job by specifying a set of statements for each function call. If you are issuing a request to shutdown the API server only, a request structure statement is not required, as illustrated in the JCL stream.

The QUIT statement should be the last statement coded in the input stream, which terminates the program.

A.8.2 JCL to Execute ENTBJAPI — BC1JAAPI

The JCL in this section is distributed with Endeavor as member name BC1JAAPI in the iprfx.igual.JCL data set. This job shows how to execute program ENTBJAPI. This program is distributed as a load module only.

The JCL to execute program ENTBJAPI appears below. Look at the JCL comment statements that describe updates you need to make before you execute this JCL stream.

In this example, DD name, MSG3FILE, describes the message response file and DD name, EXT1ELM, describes the response file for the API responses.

```
//* ( COPY JOBCARD )
//*****
//*
//* BC1JAAPI - THIS IS SAMPLE JCL TO INVOKE THE ASSEMBLER
//*          VERSION OF OUR SAMPLE API ENDEVOR APPLICATION
//*          PROGRAM: ENTBJAPI
//*
//* 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 STEPLIB AND CONLIB DATA SET NAMES ARE
//*    CORRECT.
//*    - uprfx.uqual.AUTHLIB
//*    - iprfx.igual.AUTHEXT
//*    - iprfx.igual.CONLIB
//* 3. ADD THE NECESSARY DD STATEMENTS NEEDED TO PROCESS ANY
//*    ADDITIONAL API LIST REQUESTS. CURRENTLY THIS JOB IS SETUP
//*    TO HANDLE ONLY ONE LIST REQUEST.
//* 4. MODIFY THE PROGRAMS DATA INPUT IN STEP1. AS AN EXAMPLE, THE
//*    INPUT DATA HAS A SINGLE ELEMENT EXTRACT REQUEST.
//*****
//STEP1 EXEC PGM=NDVRC1,PARM='ENTBJAPI',DYNAMNBR=1500,REGION=4096K
//STEPLIB DD DSN=uprfx.uqual.AUTHLIB,DISP=SHR
//        DD DSN=iprfx.igual.AUTHEXT,DISP=SHR
//CONLIB DD DSN=iprfx.igual.CONLIB,DISP=SHR
//SYSOUT DD SYSOUT=*
//SYSPRINT DD SYSOUT=*
//BSTERR DD SYSOUT=*
//BSTAPI DD SYSOUT=*
//MSG3FILE DD DSN=&MSG3FILE,DISP=(NEW,PASS),
//           UNIT=SYSDA,SPACE=(TRK,(5,5)),
//           DCB=(RECFM=FB,LRECL=133,BLKSIZE=13300)
//EXT1ELM DD DSN=&EXT1ELM,DISP=(NEW,PASS),
//           UNIT=SYSDA,SPACE=(TRK,(5,5)),
//           DCB=(RECFM=VB,LRECL=2048,BLKSIZE=22800)
//SYSIN DD *
```

```

* RECORD ID IS IN COLUMNS 1-5
*
*-----1-----2-----3-----4-----5-----6-----7--
***** AACTL = CONTROL STRUCTURE INFORMATION
*   V - COLUMN 6 = SHUTDOWN FLAG STRUCTURE
*   VVVVVVVV - COLUMN 7-14 IS THE MSG FILE DDNAME
*   VVVVVVVV - COLUMN 15-22 OUTPUT DATA FILE DDNAME
*
*-----1-----2-----3-----4-----5-----6-----7--
* FOR MANY REQUESTS, THE FOLLOWING SETTINGS ARE APPROPRIATE
*   V - COLUMN 6 = PATH SETTING
*   = ' ' FOR LOGICAL
*   = 'L' FOR LOGICAL
*   = 'P' FOR PHYSICAL
*   V - COLUMN 7 = RETURN SETTING
*   = ' ' FOR FIRST FOUND
*   = 'F' FOR FIRST FOUND
*   = 'A' FOR ALL FOUND
*   V - COLUMN 8 = SEARCH SETTING
*   = ' ' FOR FIRST
*   = 'A' FOR ALL
*   = 'B' FOR BETWEEN
*   = 'E' FOR NEXT
*   = 'N' FOR NO
*   = 'R' FOR RANGE
*   V - COLUMN 9 = UNUSED
*
*-----1-----2-----3-----4-----5-----6-----7--
***** ALENV = LIST ENVIRONMENT STRUCTURE INFORMATION
*   V - COLUMN 6 = PATH SETTING
*   V - COLUMN 7 = RETURN SETTING
*   V - COLUMN 8 = SEARCH SETTING
*   V - COLUMN 9 = UNUSED
*   VVVVVVVV - COLUMN 10-17 ENVIRONMENT NAME
*   V - COLUMN 18 = STAGE ID
*   VVVVVVVV - COLUMN 19-26 = TO ENVIRONMENT NAME
*   V - COLUMN 27 = TO STAGE ID
* NOTE: IF BETWEEN/RANGE SETTINGS ARE USED, YOU NEED TO SPECIFY
*   TO-ENV AND TO-STAGE, OTHERWISE LEAVE BLANK.
*-----1-----2-----3-----4-----5-----6-----7--
***** ALSTG = LIST STAGE STRUCTURE INFORMATION
*   V - COLUMN 6 = PATH SETTING
*   V - COLUMN 7 = RETURN SETTING
*   V - COLUMN 8 = SEARCH SETTING
*   V - COLUMN 9 = UNUSED
*   VVVVVVVV - COLUMN 10-17 ENVIRONMENT NAME
*   V - COLUMN 18 = STAGE ID
*   VVVVVVVV - COLUMN 19-26 = TO ENVIRONMENT NAME
*   V - COLUMN 27 = TO STAGE ID
* NOTE: IF BETWEEN/RANGE SETTINGS ARE USED, YOU NEED TO SPECIFY
*   TO-ENV AND TO-STAGE, OTHERWISE LEAVE BLANK.
*-----1-----2-----3-----4-----5-----6-----7--

```

```

***** ALSYS = LIST SYSTEM STRUCTURE INFORMATION
*   V - COLUMN 6 = PATH SETTING
*   V - COLUMN 7 = RETURN SETTING
*   V - COLUMN 8 = SEARCH SETTING
*   V - COLUMN 9 = UNUSED
*   VVVVVVVV - COLUMN 10-17 ENVIRONMENT NAME
*   V - COLUMN 18 = STAGE ID
*   VVVVVVVV - COLUMN 19-26 SYSTEM NAME
*   VVVVVVVV - COLUMN 27-34 = TO ENV NAME
*   V - COLUMN 35 = TO STAGE ID
* NOTE: IF BETWEEN/RANGE SETTINGS ARE USED, YOU NEED TO SPECIFY
*       TO-ENV AND TO-STAGE, OTHERWISE LEAVE BLANK.
*---+---1---+---2---+---3---+---4---+---5---+---6---+---7---
***** ALSBS = LIST SUBSYSTEM STRUCTURE INFORMATION
*   V - COLUMN 6 = PATH SETTING
*   V - COLUMN 7 = RETURN SETTING
*   V - COLUMN 8 = SEARCH SETTING
*   V - COLUMN 9 = UNUSED
*   VVVVVVVV - COLUMN 10-17 ENVIRONMENT NAME
*   V - COLUMN 18 = STAGE ID
*   VVVVVVVV - COLUMN 19-26 SYSTEM NAME
*   VVVVVVVV - COLUMN 27-34 SUBSYSTEM NAME
*   VVVVVVVV - COLUMN 35-42 = TO ENV NAME
*   V - COLUMN 43 = TO STAGE ID
*---+---1---+---2---+---3---+---4---+---5---+---6---+---7---
***** ALSIT = LIST SITE STRUCTURE INFORMATION
*---+---1---+---2---+---3---+---4---+---5---+---6---+---7---
***** ALELM = LIST ELEMENT STRUCTURE INFORMATION
*   V - COLUMN 6 = PATH SETTING
*   V - COLUMN 7 = RETURN SETTING
*   V - COLUMN 8 = SEARCH SETTING
*   V - COLUMN 9 = UNUSED
*   VVVVVVVV - COLUMN 10-17 ENVIRONMENT NAME
*   V - COLUMN 18 = STAGE ID
*   VVVVVVVV - COLUMN 19-26 SYSTEM NAME
*   VVVVVVVV - COLUMN 27-34 SUBSYSTEM NAME
*   COLUMN 35-44 = ELEMENT NAME VVVVVVVVVV
*   COLUMN 45-52 = TYPE NAME VVVVVVVV
*   COLUMN 53-60 = TO-ENV NAME VVVVVVVV
*   COLUMN 61 = TO-STAGE ID V
*   COLUMN 62-71 = THRU-ELEMENT NAME VVVVVVVVVV
* NOTE: IF BETWEEN/RANGE SETTINGS ARE USED, YOU NEED TO SPECIFY
*       TO-ENV AND TO-STAGE, OTHERWISE LEAVE BLANK.

```

```

*-----1-----2-----3-----4-----5-----6-----7--
***** ALPGR = LIST PROCESSOR GROUP STRUCTURE INFORMATION
*   V - COLUMN 7 = RETURN SETTING
*   VVVVVVVV - COLUMN 10-17 ENVIRONMENT NAME
*   V - COLUMN 18 = STAGE ID
*   VVVVVVVV - COLUMN 19-26 SYSTEM NAME
*   VVVVVVVV - COLUMN 27-34 TYPE NAME
*   VVVVVVVV - COLUMN 35-42 PROC GROUP
*-----1-----2-----3-----4-----5-----6-----7--
***** ALDSN = LIST DATA SET STRUCTURE INFORMATION
*   V - COLUMN 7 = RETURN SETTING
*   VVVVVVVV - COLUMN 10-17 ENVIRONMENT NAME
*   V - COLUMN 18 = STAGE ID
*   VVVVVVVV - COLUMN 19-26 SYSTEM NAME
*   VV - COLUMN 27-28 DATA SET ID
*-----1-----2-----3-----4-----5-----6-----7--
***** ALAGR = LIST APPROVER GROUP STRUCTURE INFORMATION
*   V - COLUMN 7 = RETURN SETTING
*   VVVVVVVV - COLUMN 10-17 ENVIRONMENT NAME
*   VVVVVVVV - COLUMN 18-25 APPROVER GROUP NAME
*-----1-----2-----3-----4-----5-----6-----7--
***** ALAGJ = LIST APPROVER JUNCTION STRUCTURE INFORMATION
*   V - COLUMN 7 = RETURN SETTING
*   VVVVVVVV - COLUMN 10-17 ENVIRONMENT NAME
*   VVVVVVVVVVVVVVVVVVVVVVVVVVVVV - COLUMN 18-42 APPR JUNCTION
*   VVVVVVVV - COLUMN 18-25 SYSTEM
*   VVVVVVVV - COLUMN 26-33 SUBSYSTEM
*   VVVVVVVV - COLUMN 34-41 TYPE
*   V - COLUMN 34-41 STAGE NUM
*-----1-----2-----3-----4-----5-----6-----7--
***** AEELM = EXTRACT ELEMENT STRUCTURE INFORMATION
*   V - COLUMN 6 = FORMAT SETTING
*   = ' ' FOR NO FORMAT, JUST EXTRACT ELEMENT
*   = 'B' FOR ENDEVOR BROWSE DISPLAY FORMAT
*   = 'C' FOR ENDEVOR CHANGE DISPLAY FORMAT
*   = 'H' FOR ENDEVOR HISTORY DISPLAY FORMAT
*   V - COLUMN 7 = RECORD TYPE SETTING
*   = 'E' FOR ELEMENT
*   = 'C' FOR COMPONENT
*   VVVVVVVV - COLUMN 10-17 ENVIRONMENT NAME
*   V - COLUMN 18 = STAGE ID
*   VVVVVVVV - COLUMN 19-26 SYSTEM NAME
*   VVVVVVVV - COLUMN 27-34 SUBSYSTEM NAME
*   COLUMN 35-44 = ELEMENT NAME VVVVVVVVVV
*   COLUMN 45-52 = TYPE NAME VVVVVVVV
*   COLUMN 53-54 = VERSION VV
*   COLUMN 55-56 = LEVEL VV

```

```
*---+---1---+---2---+---3---+---4---+---5---+---6---+---7---
* EXTRACT AN ELEMENT WITH NO FORMAT - CURRENT VERSION & LEVEL
AACTL MSG3FILEEXT1ELM
AEELM E INT      1NDVRMVS BASE      BC1PFPVL  ASMPGM
RUN
* LAST CALL, ENSURE THAT THE API SERVER IS SHUTDOWN.
AACTLY
RUN
QUIT
//*
//* PRINT ANY MESSAGES
//STEP2  EXEC PGM=IEBGENER
//SYSPRINT DD DUMMY
//SYSIN   DD DUMMY
//SYSUT1  DD DSN=&&MSG3FILE,DISP=(OLD,DELETE)
//SYSUT2  DD SYSOUT=*
//*
//* PRINT EXTRACTED ELEMENT
//STEP3  EXEC PGM=IEBGENER
//SYSPRINT DD DUMMY
//SYSIN   DD DUMMY
//SYSUT1  DD DSN=&&EXT1ELM,DISP=(OLD,DELETE)
//SYSUT2  DD SYSOUT=*
```

Appendix B. Index

Special Characters

"From" Endeavor location, last 2-38
* 1-22
% 1-22

A

AACTL control structure fields 2-3
AACTL_MSG_DDN file 3-3
AACTL_REASON field 1-17
AACTL_RTNCODE field 1-17
Actions
 add element 2-11
 build generate SCL 2-107
 delete element 2-13
 generate element 2-24
 last for list element 2-34
 list
 element options 2-27
 package summary 2-73
 move element 2-42
 print
 element 2-44
 member element 2-46
 retrieve element 2-47
 sample for elements 3-4
 signin element 2-49
 submit package 2-98
 transfer element 2-51
 update element 2-54
Add
 data set, last 2-36
 element action 2-11
AEADD_RQ request structure fields 2-11

AEDEL_RQ request structure fields 2-13
AEELM_RQ request structure fields 2-16
AEELM_RS response structure fields 2-17
AEGEN_RQ request structure fields 2-24
AEMOV_RQ request structure fields 2-42
AEPRE_RQ request structure fields 2-44
AEPRM_RQ request structure fields 2-46
AERET_RQ request structure fields 2-47
AESIG_RQ request structure fields 2-49
AETRA_RQ request structure fields 2-51
AEUPD_RQ request structure fields 2-54
ALAGJ_RQ request structure fields 2-103
ALAGJ_RS response structure fields 2-104
ALAGR_RQ request structure fields 2-101
ALAGR_RS response structure fields 2-101
ALAPP_RQ request structure fields 2-78
ALBKO_RQ request structure fields 2-80
ALBKO_RS request structure fields 2-80
ALCAS_RQ request structure fields 2-82
ALCAS_RS response structure fields 2-82
ALCMP_RQ request structure fields 2-105
ALCMP_RS response structure fields 2-108
ALCOR_RQ request structure fields 2-83
ALCOR_RS response structure fields 2-83
ALDIR_RQ request structure fields 2-113
ALDIR_RS response structure fields 2-114
ALDSN_RQ request structure fields 2-111
ALDSN_RS response structure fields 2-112
ALELB_RS fields 2-39
ALELM_RQ request structure fields 2-26
ALELM_RS response structure fields 2-32
ALELX_RS fields 2-41
ALENV_RQ request structure fields 2-116
ALENV_RS response structure fields 2-117

AllFusion Harvest CM packages 2-56

ALPGR_RQ request structure fields 2-119

ALPGR_RS response structure fields 2-120

ALPKB_RS response structure fields 2-92

ALPKG_RQ request structure fields 2-85

ALPKG_RS response structure fields 2-89

ALSBS_RQ request structure fields 2-128

ALSBS_RS response structure fields 2-129

ALSCL_RQ request structure fields 2-94

ALSCL_RS response structure fields 2-94

ALSIT_RQ request structure fields 2-122

ALSIT_RS response structure fields 2-122

ALSTG_RQ request structure fields 2-126

ALSTG_RS response structure fields 2-127

ALSUM_RQ request structure fields 2-73

ALSUM_RS response structure fields 2-73

ALSYS_RQ request structure fields 2-131

ALSYS_RS response structure fields 2-132

ALTYP_RQ request structure fields 2-134

ALTYP_RS response structure fields 2-135

APAPP_RQ request structure fields 2-60

APBKI_RQ request structure fields 2-61

APBKO_RQ request structure fields 2-62

APCAS_RQ request structure fields 2-63

APCOM_RQ request structure fields 2-66

APCOR_RQ request structure fields 2-96

APDEF_RQ request structure fields 2-67

APDEL_RQ request structure fields 2-70

APDEN_RQ request structure fields 2-71

APEXE_RQ request structure fields 2-72

API Diagnostic Trace 3-2

API Internal Trace 3-2

API\$INIT macro 1-12

Approve package 2-60

Approver group, list 2-101

Approvers, list package 2-78

APSUB_RQ request structure fields 2-98

Architecture of API 1-4

Assembler

- calling API from 1-15
- initializing programs 1-12
- macros
 - about 1-7
 - ENHAAREB 2-10
 - ENHAEADD 2-11
 - ENHAEDEL 2-13
 - ENHAEELM 2-15

Assembler (*continued*)

macros (*continued*)

- ENHAEGEN 2-24
- ENHAEMOV 2-42
- ENHAEPRE 2-44
- ENHAEPRM 2-46
- ENHAERET 2-47
- ENHAESIG 2-49
- ENHAETRA 2-51
- ENHAEUPD 2-54
- ENHALAGJ 2-103
- ENHALAGR 2-101
- ENHALAPP 2-78
- ENHALBKO 2-80
- ENHALCAS 2-82
- ENHALCMP 2-105
- ENHALCOR 2-83
- ENHALDIR 2-113
- ENHALDSN 2-111
- ENHAELM 2-26
- ENHAENV 2-116
- ENHALPGR 2-119
- ENHALPKG 2-85
- ENHALSBS 2-128
- ENHALSCL 2-94
- ENHALSIT 2-122
- ENHALSTG 2-126
- ENHALSUM 2-73
- ENHALSYS 2-131
- ENHALTYP 2-134
- ENHAPAPP 2-60
- ENHAPBKI 2-61
- ENHAPBKO 2-62
- ENHAPCAS 2-63
- ENHAPCOM 2-66
- ENHAPCOR 2-96
- ENHAPDEF 2-67
- ENHAPDEL 2-70
- ENHAPDEN 2-71
- ENHAPEXE 2-72
- ENHAPRES 2-97
- ENHAPSUB 2-98

sample programs A-10

B

Backin Package 2-61
Backout
 list packages 2-80
 package 2-62
Base
 element 2-34
 structure 2-31
BC1JAAPI JCL A-19
BC1JAPGM JCL A-10
BC1JEPGM JCL A-12
BC1JPLST JCL A-15
BC1JPUPD JCL A-17
BC1JRAPI JCL A-7
BC1PAPI diagnostic trace 3-9
Browse record layout
 component extract 2-21
 element extract 2-19
Build
 generate action SCL 2-107
 list directory 2-113

C

C1MSGS1 DD statement 2-69, 3-3
Calling API 1-15
Calls, function 2-8
Cast
 list packages 2-82
 packages 2-63
CCID 2-13, 2-24, 2-44, A-5
Change record layout
 component extract 2-22
 element extract 2-19
COBOL
 calling API from 1-16
 CCIDRPT1 sample program A-5
 copybooks
 about 1-7
 ECHAAREB 2-10
 ECHAEADD 2-11
 ECHAEDDEL 2-13
 ECHAEELM 2-15
 ECHAEGEN 2-24
 ECHAEMOV 2-42
 ECHAEPRE 2-44
 ECHAEPRM 2-46

COBOL (continued)

 copybooks (continued)
 ECHAERET 2-47
 ECHAESIG 2-49
 ECHAETRA 2-51
 ECHAEUPD 2-54
 ECHALAGJ 2-103
 ECHALAGR 2-101
 ECHALAPP 2-78
 ECHALBKO 2-80
 ECHALCAS 2-82
 ECHALCMP 2-105
 ECHALCOR 2-83
 ECHALDIR 2-113
 ECHALDSN 2-111
 ECHALELM 2-26
 ECHALENV 2-116
 ECHALPGR 2-119
 ECHALPKG 2-85
 ECHALSBS 2-128
 ECHALSCL 2-94
 ECHALSIT 2-122
 ECHALSTG 2-126
 ECHALSUM 2-73
 ECHALSYS 2-131
 ECHALTYP 2-134
 ECHAPAPP 2-60
 ECHAPBKI 2-61
 ECHAPBKO 2-62
 ECHAPCAS 2-63
 ECHAPCOM 2-66
 ECHAPCOR 2-96
 ECHAPDEF 2-67
 ECHAPDEL 2-70
 ECHAPDEN 2-71
 ECHAPEXE 2-72
 ECHAPRES 2-97
 ECHAPSUB 2-98
 field names 1-12
 initializing structures in 1-12
 templates for function calls 1-16
Codes
 debugging help 3-2
 reason 1-17
 return 1-17

Comment location data for list components 2-106
Commit package 2-66
Components
 data 2-15
 list/where-used 2-105
 lists 2-35
 validate 2-63
CONAPI Utility 1-4
Control structure 1-5, 2-3
Copybooks, COBOL 1-7
Correlation
 list package 2-83
 package 2-96
Create packages 2-67
CSECT build list 2-113

D

Data
 available for list package locations 2-74
 comment location for list components 2-106
 element location for list components 2-105
 extract component 2-15
 member location for list components 2-106
 object location for list components 2-106
 set
 last add or update 2-36
 list 2-111
 trace 3-2
Date formats, response structure 1-6
DD statements 3-3
Debugging help 3-2, 3-10
Define package
 function call 2-67
 sample report 3-3
Delete
 element action 2-13
 packages 2-70
Delta
 component list 2-35
 element 2-34
Deny packages 2-71
Diagnostic Trace 3-2, 3-9
Directory list 2-113

E

ECHAACTL initializing 1-12
Elements
 action sample report 3-4
 actions not supported 1-2
 add 2-11
 base 2-34
 delete 2-13
 delta (last level) 2-34
 extract 2-15
 generate 2-24
 last move 2-36
 list 2-26
 location data for list components 2-105
 long names 2-10
 move action 2-42
 package last executed against 2-38
 print
 action 2-44
 member action 2-46
 processor execution 2-37
 record layouts 2-17
 retrieve
 action 2-47
 last 2-37
 sample function call A-12
 signin action 2-49
 transfer action 2-51
 update 2-54
EN\$TRAPI internal trace 3-10
ENAS\$NDVR 1-4
Endevor location, last "from" 2-38
ENHAACTL initializing 1-12
ENHAAPGM sample program A-10
ENHAEPGM A-12
ENHAPLST A-15
ENHAPUPD A-17
ENTBJAPI A-19
Enterprise package function 2-56
Environment list
 function call 2-116
 sample A-10
Execute
 CCIDRPT1 A-7
 ENHAAPGM A-10

Execute (*continued*)

ENHAEPGM A-12

ENHAPLST A-15

ENHAPUPD A-17

ENTBJAPI A-19

packages 2-72

Executing programs A-3

Execution

element processor 2-37

reports 3-3

window 2-63

Extension

records 2-31

request 2-10

Extract

component data 2-15

elements 2-15

function calls 1-6

list approver group information 2-103

Extraction types 2-15

F

Facilities, trace 3-2

Facility, diagnostic trace 3-9

Fields

AACTL control structure 2-3

AACTL_REASON 1-17

AACTL_RTNCODE 1-17

AEADD_RQ request structure 2-11

AEDEL_RQ request structure 2-13

AEELM_RQ request structure 2-16

AEELM_RS response structure 2-17

AEGEN_RQ request structure 2-24

AEMOV_RQ request structure 2-42

AEPRE_RQ request structure 2-44

AEPRM_RQ request structure 2-46

AERET_RQ request structure 2-47

AESIG_RQ request structure 2-49

AETRA_RQ request structure 2-51

AEUPD_RQ request structure 2-54

ALAGJ_RQ request structure 2-103

ALAGJ_RS response structure 2-104

ALAGR_RQ request structure 2-101

ALAGR_RS response structure 2-101

ALAPP_RQ request structure 2-78

Fields (*continued*)

ALBKO_RQ request structure 2-80

ALBKO_RS request structure 2-80

ALCAS_RQ request structure 2-82

ALCAS_RS response structure 2-82

ALCMP_RQ request structure 2-105

ALCMP_RS response structure 2-108

ALCOR_RQ request structure 2-83

ALCOR_RS response structure 2-83

ALDIR_RQ request structure 2-113

ALDIR_RS response structure 2-114

ALDSN_RQ request structure 2-111

ALDSN_RS response structure 2-112

ALELB_RS 2-39

ALELM_RQ request structure 2-26

ALELM_RS response structure 2-32

ALELX_RS 2-41

ALENV_RQ request structure 2-116

ALENV_RS response structure 2-117

ALPGR_RQ request structure 2-119

ALPGR_RS response structure 2-120

ALPKB_RS response structure 2-92

ALPKG_RQ request structure 2-85

ALPKG_RS response structure 2-89

ALSBS_RQ request structure 2-128

ALSBS_RS response structure 2-129

ALSCL_RQ request structure 2-94

ALSCL_RS response structure 2-94

ALSIT_RQ request structure 2-122

ALSIT_RS response structure 2-122

ALSTG_RQ request structure 2-126

ALSTG_RS response structure 2-127

ALSUM_RQ request structure 2-73

ALSUM_RS response structure 2-73

ALSYS_RQ request structure 2-131

ALSYS_RS response structure 2-132

ALTYP_RQ request structure 2-134

ALTYP_RS response structure 2-135

APAPP_RQ request structure 2-60

APBKI_RQ request structure 2-61

APBKO_RQ request structure 2-62

APCAS_RQ request structure 2-63

APCOM_RQ request structure 2-66

APCOR_RQ request structure 2-96

APDEF_RQ request structure 2-67

APDEL_RQ request structure 2-70

Fields (*continued*)

- APDEN_RQ request structure 2-71
- APEXE_RQ request structure 2-72
- APSUB_RQ request structure 2-98
- enterprise package 2-56
- Note for APDEF_RQ 2-69

Files

- AACTL_MSG_DDND 3-3
- HFS 2-10
- message 1-18, 3-3
- writing responses to output 1-19

Filters, output for list components 2-107

Formats

- for dates in response structure 1-6
- rectyp for list components 2-109

From structure 2-31

Full structure 2-31

Function

- calls
 - about 2-8
 - element action samples A-12
 - list of 1-7
 - storing responses 1-6
- enterprise package 2-56

G

Generate

- action SCL 2-107
- elements 2-24

Group

- junctions, list approver 2-103
- list
 - approver 2-101
 - processor 2-119

H

Header, list package 2-85

HFS files 2-10, 2-113

History record layout

- component extract 2-23
- element extract 2-20

I

Initializing API structures 1-12

Internal Trace 3-2, 3-10

Inventory list sample

- function call A-19
- report 3-5

ISPF package panels 2-58

J

JCL

- BC1JAAPI A-19
- BC1JAPGM A-10
- BC1JEPGM A-12
- BC1JPLST A-15
- BC1JPUPD A-17
- CCIDRPT1 A-7

Junctions, list approver group 2-103

L

Last

- "from" Endeavor location 2-38
- action for list element 2-34
- add data set 2-36
- element move 2-36
- level (element delta) 2-34
- retrieved element 2-37
- update data set 2-36

Layouts, record 2-17

List

- approver group 2-101
- component 2-35
- components/where-used 2-105
- data set 2-111
- directory 2-113
- elements 2-26
- environment
 - function call 2-116
 - sample program A-10
- inventory
 - function call sample A-19
 - sample report 3-5
- package
 - action summary 2-73
 - approvers 2-78
 - backout 2-80

List (*continued*)

- package (*continued*)
 - cast report 2-82
 - correlation 2-83
 - function call 2-73
 - header 2-85
 - sample function call A-15
 - SCL 2-94
- processor group 2-119
- sample reports for packages 3-5
- site 2-122
- stage 2-126
- subsystem 2-128
- system 2-131
- type 2-134

Location

- data for list components 2-105
- last "from" Endeavor 2-38
- list package action summary, source 2-74

Logical mapping requests 2-8

Long names 2-10

M

Macros

- API\$INIT 1-12
- Assembler 1-7

Map searching

- ALAGJ_RQ 2-104
- ALAGR_RQ request structure 2-101

Mapping requests 2-8

Masking, name 1-22

Master structure 2-31

Member

- element action, print 2-46
- location data for list components 2-106

Messages

- file 1-18, 3-3

Modify packages 2-67

Move

- element action 2-42
- last element 2-36

N

Name masking 1-22

Naming conventions structures 1-5

NDVRC1 program A-3

No format record layout 2-18

Not supported actions 1-2

Note fields for APDEF_RQ 2-69

O

Object location data for list components 2-106

Options, list element action 2-27

Output

- file, writing responses to 1-19
- filters for list components 2-107

Overrides 2-119

P

Packages

- actions not supported 1-2

AllFusion Harvest CM 2-56

approve 2-60

approvers, list 2-78

backin 2-61

backout

- function call 2-62

- list function call 2-80

cast

- function call 2-63

- report, list 2-82

commit 2-66

correlation

- function call 2-96

- list function call 2-83

define

- function call 2-67

- sample 3-3

delete 2-70

deny 2-71

enterprise function 2-56

execute 2-72

header, list 2-85

last executed against element 2-38

list action

- sample A-15

- summary 2-73

reset 2-97

Packages (*continued*)

- sample report for list 3-5
- SCL, list 2-94
- submit request 2-98
- update
 - function call sample A-17
 - sample report 3-7

Panels

- enterprise package 2-56
- ISPF package 2-58

Physical mapping requests 2-8

Placeholder 1-22

Print

- element action 2-44
- member element action 2-46

Processor

- execution, element 2-37
- list group 2-119

Programs

- CCIDRPT1 COBOL A-5
- ENASNDVR 1-4
- executing A-3
- initializing Assembler 1-12
- NDVRC1 A-3

R

Reason codes 1-17

Record

- extension 2-31
- layouts 2-17

Rectyp formats for list components 2-109

Reports

- CCIDRPT1 sample program A-6
- define package sample 3-3
- element action sample 3-4
- execution 3-3
- inventory list sample 3-5
- list package cast 2-82
- sample for list package 3-5
- update package sample 3-7

Request

- extension 2-10, 2-113
- mapping 2-8
- structure 1-6
- submit package 2-98

Request Extension

- structure 1-6

Reset package 2-97

Response structure 1-6

Restrictions, enterprise package 2-58

Retrieve

- element
 - action 2-47
 - data 2-15
 - last element 2-37

Return codes 1-17

S

Samples

- CCIDRPT1 COBOL program A-5
- diagnostic trace 3-9
- element action function call A-12
- EN\$TRAPI internal trace 3-10
- ENHAAPGM A-10
- inventory list function call A-19
- list package action function call A-15
- reports 3-3
- update package function call A-17

SCL

- build generate action 2-107
- fields 2-67
- list package 2-94

Searching

- ALAGJ_RQ 2-104
- ALAGR_RQ request structure 2-101
- list
 - elements 2-27
 - environment 2-117
 - stage 2-127
 - subsystem 2-129
 - system 2-132
 - type 2-135

Selection criteria for list elements 2-29

Shutting down API server 1-14

Signin element action 2-49

Site, list 2-122

Source location list package action summary 2-74

Stage, list 2-126

Starting API server 1-14

Statement, CIMSGS1 3-3
Storing function call responses 1-6
Structures
 about 1-5
 control 1-5, 2-3
 list element 2-31
 naming conventions 1-5
 request 1-6
 request extension 1-6
 response 1-6
Submit package request 2-98
Subsystem, list 2-128
Summary, list package 2-73
Symbolic overrides 2-119
System, list 2-131

T

Target
 location for list package summary 2-75
 structure 2-31
Templates for COBOL function calls 1-16
Trace
 diagnostic 3-9
 facilities 3-2
 internal 3-10
Transfer element action 2-51
Type, list 2-134
Types, extraction 2-15

U

Unsupported actions 1-2
Update
 data set, last 2-36
 elements 2-54
 package sample
 function call A-17
 report 3-7
Using
 placeholders 1-22
 wildcards 1-22
Utility, CONAPI 1-4

V

Validate components 2-63

W

Where-used, list components 2-105
Wildcard 1-22
Window, execution 2-63
Writing
 messages to message file 1-18
 responses to response file 1-19

