

Note!

Before using this information and the product it supports, be sure to read the general information under "Notices" on page v.

First Edition (December 1995)

This edition applies to Version 3, Release 6, Modification Level 0, of Application Development Manager/400 (Feature 2213), a feature of IBM Application Development ToolSet/400 (Program 5716-PW1), and to all subsequent releases and modifications until otherwise indicated in new editions. Make sure you are using the correct edition for the level of the product.

Order publications through your IBM representative or the IBM branch office serving your locality. Publications are not stocked at the address given below.

A form for readers' comments is provided at the back of this publication. If the form has been removed, address your comments to:

IBM Canada Ltd. Laboratory
Information Development
2G/345/1150/TOR
1150 Eglinton Avenue East
North York, Ontario, Canada. M3C 1H7

You can also send your comments by facsimile (attention: RCF Coordinator), or you can send your comments electronically to IBM. See "Communicating Your Comments to IBM" for a description of the methods. This page immediately precedes the Readers' Comment Form at the back of this publication.

When you send information to IBM, you grant IBM a nonexclusive right to use or distribute the information in any way it believes appropriate without incurring any obligation to you.

© **Copyright International Business Machines Corporation 1992, 1995. All rights reserved.**

Note to U.S. Government Users — Documentation related to restricted rights — Use, duplication or disclosure is subject to restrictions set forth in GSA ADP Schedule Contract with IBM Corp.

Contents

Notices	v
Trademarks and Service Marks	v
Chapter 1. Introducing the Application Development Manager/400 Application Programming Interface	1
Chapter 2. The Program QLYALCPR	3
The Call Interface	4
Data Structures	6
Error Messages and Return Codes	7
Examples of How to Use the API	8
Chapter 3. The Program QLYGSPTH	13
The Call Interface	13
Error Messages and Return Codes	14
Examples of How to Use the API	17
Chapter 4. The Program QLYVUDPT	19
The Call Interface	19
Error Messages and Return Codes	20
Example of How to Use the API	20
Chapter 5. The External Data Files	21
The Project Registration File	21
The Group Hierarchy File	22
The Part Directory Files	23
Index	29

Notices

Any reference to an IBM licensed program in this publication is not intended to state or imply that only IBM's licensed program may be used. Any functionally equivalent product, program or service that does not infringe any of IBM's intellectual property rights may be used instead of the IBM product, program, or service. Evaluation and verification of operation in conjunction with other products, except those expressly designated by IBM, is the user's responsibility.

IBM may have patents or pending patent applications covering subject matter in this document. The furnishing of this document does not give you any license to these patents. You can send license inquiries, in writing, to the IBM Director of Licensing, IBM Corporation, 500 Columbus Avenue, Thornwood, NY 10594, USA.

Licensees of this program who wish to have information about it for the purpose of enabling: (i) the exchange of information between independent created programs and other programs (including this one) and (ii) the mutual use of the information which has been exchanged, should contact IBM Canada Ltd., Department 071, 1150 Eglinton Avenue East, North York, Ontario M3C 1H7, Canada. Such information may be available, subject to appropriate terms and conditions, including in some cases payment of a fee.

This publication contains examples of data and reports used in daily business operations. To illustrate them as completely as possible, the examples include the names of individuals, companies, brands, and products. All of these names are fictitious and any similarity to the names and addresses used by an actual business enterprise is entirely coincidental.

Trademarks and Service Marks

The following terms are trademarks of International Business Machines Corporation in the United States or other countries or both:

Application System/400	IBM
AS/400	IBMLink
C/400	400

Chapter 1. Introducing the Application Development Manager/400 Application Programming Interface

This application programming interface (API) for Application Development Manager/400 a feature of the Application Development ToolSet/400 (ADTS/400) product, is provided so that application programmers can write their own interface to this feature. The complete API consists of the following items:

- The callable program, QLYALCPR, which allocates a project so that while one developer is working with a project, no other developer can alter it
- The callable program, QLYGSPTH, which retrieves a search path for a project starting at the specified group
- The callable program, QLYVUDPT, which determines if a part type is user-defined, and if the user-defined part type is stored in a source file member
- A set of four files that give application developers the ability to build lists
- A set of CL commands provided by the Application Development Manager/400 feature that the application programmer uses to write the interface

This document assumes that its readers are familiar with the Application Development Manager/400 feature. It describes only the API's callable programs and four external files. For information about the CL commands, refer to the following publications provided with the Application Development Manager/400 feature.

- *ADTS/400: Application Development Manager/400 Introduction and Planning Guide, GC09-1807*
- *ADTS/400: Application Development Manager/400 User's Guide, SC09-2133*

Chapter 2. The Program QLYALCPR

The program QLYALCPR ensures that a project's hierarchy and part lists remain consistent while a developer is working with any group in a project. It is intended to prevent a situation where two developers could be working with the same project, for example, one developer working through the WRKPARTPDM command and another developer working through the CHGGRP command. This situation could lead to problems similar to those described in the following scenarios:

1. Developer A is working with a list of parts using the WRKPARTPDM command. The list is built one page at a time. After the first page has been built, developer B uses the CHGGRP command to change a group whose parts are in developer A's list.

Developer A then presses the Page Down key to display the next page. Because the underlying hierarchy has changed, the second page of parts is now inconsistent with the first page. If developer A carries out an operation on a part on the first page by typing the number of the desired option, the operation will likely end in error.

2. Developer A is working with a list of parts using the WRKPARTPDM command, as described above. However, this time developer B is changing the group through the WRKGRPPDM command. The same problem occurs when developer A presses the Page Down key.

You can use the API to prevent such problems in one of three ways:

1. To check what authority a user has to a particular project.
2. To allocate and deallocate a project when a developer is working with parts in a project.
3. To obtain a list of projects to which a user is authorized in a user space.

The Call Interface

This program or module, called QLYALCPR to signify *allocate project*, is shipped in the library QADM as part of the Application Development Manager/400 licensed program. It is built as a user-domain program that is allowed to run in system state. You must have the Application Development Manager/400 feature installed to access this program.

The call has the following format:

```
CALL PGM(QLYALCPR) PARM(PROJECT ACTION USRSPC STATUS ERRCODE)
```

Table 1. The parameters for QLYALCPR

Parameter	Use	Type	Description
PROJECT	Input	CHAR(32)	Allowed values—alphanumeric character string
ACTION	Input	CHAR(10)	Allowed values—*CHK, *ALC, *DLC, *LIST
USRSPC	Input	CHAR(20)	Allowed values—10-character space name, followed by 10-character library name. *LIBL and *CURLIB are allowed for library.
STATUS	Output	CHAR(10)	Allowed values—*MRKDLT, *RCL, *NORMAL, *ERROR
ERRCODE	Input/Output	CHAR(*)	This is a mandatory parameter for the Application System/400 (AS/400) APIs. Its structure is explained in more detail in “The ERRCODE parameter” on page 6.

Before a developer can work with parts in a particular project, this API must allocate the project. If the developer is not authorized to the project, or another developer is using the project, or the project does not exist, the appropriate error message is returned. See “Error Messages and Return Codes” on page 7 for a list of these error messages. If no error message is issued, the operation completed successfully.

A status variable is also returned, indicating that the project has been marked for deletion, requires a reclaim operation, is in a usable state, or that an error message has been issued.

The *PROJECT* parameter: A 32-character string that accepts a valid Application Development Manager/400 project name which is validated by the API.

The *ACTION* parameter: This parameter and the project name form the input to the API. This parameter consists of four values:

1. A check action (*CHK) that does all the error checking but does *not* do the physical allocation of the project
2. An allocate action (*ALC) that does the checking *and* the allocation
3. A deallocate action (*DLC) that deallocates a previously allocated project
4. A list action (*LIST) that returns a list of projects in a specified user space

A check action can be issued against any project at any time. An allocate action must eventually be followed by a deallocate action, otherwise the project remains allocated (or locked), even after the user has finished using the project. The lock is

released when the user signs off or the job is complete. However, it is recommended that users deallocate projects when they have finished with them. While you have a project allocated through this API, other users can issue Application Development Manager/400 commands against the project. They *cannot*, however, issue the following commands because these commands are capable of changing the underlying hierarchies and structures of a project.

- Create Group (CRTGRP)
- Change Group (CHGGRP)
- Delete Group (DLTGRP)
- Create Project (CRTPRJ)
- Change Project (CHGPRJ)
- Delete Project (DLTPRJ)
- Reclaim Project (RCLPRJ)

The list action returns a list of projects to which users are authorized and their associated descriptive texts stored in a *USRSPC object.

The USRSPC parameter: Required when the *LIST option is selected. The first 10 characters make up the user-space name, and the second 10 characters define the library containing the user space. *CURLIB and *LIBL are accepted as library values, and the error messages listed on page 8 are issued against the user space. You must create the user-space object and delete the space when you are finished with it.

The STATUS parameter: Returns a value of *MRKDLT, *RCL, *NORMAL, or *ERROR. These values can be used to restrict the options that can be run against certain projects in certain states.

The value *MRKDLT means that the project is in an incomplete state because the CRTPRJ or DLTPRJ command failed to complete successfully. As a result, only the DLTPRJ or RCLPRJ command can be run against this project. The value *RCL means that the project is in a state that requires the RCLPRJ command to be run against it before any other commands can be used against it. The value *NORMAL means that the project can be used normally. The value *ERROR means that an error message was issued while checking the project.

The *ERRCDE* parameter: A mandatory parameter with a structure that is standard to all the Application System/400 APIs.

*Table 2. The structure of the *ERRCDE* parameter*

Use	Type	Description
Input	BINARY(4)	Bytes provided The length of the area provided for the error code. If zero, an exception is signalled when an error is detected. If nonzero, the error code parameter is filled in with the exception information and no exception is signalled. It must be at least 8 if not zero. If a value for this field is specified that is not valid, that is, a negative number or a number between 0 and 8, the message CPF3CF1 is sent.
Output	BINARY(4)	Bytes available (zero if no errors) The length of data available associated with an error condition. If zero, no error was detected. The amount of data returned is the lesser of bytes provided and bytes available.
Output	CHAR(7)	Exception ID
Output	CHAR(1)	Reserved
Output	CHAR(*)	Exception data

Data Structures

The structure of the linked list returned in the user space when an action of *LIST is specified as follows.

Table 3. The project list structure

Use	Type	Description
Output	CHAR(32)	Project name—the name of the project the user is authorized to.
Output	CHAR(80)	Project text description—the text description of the authorized project.
Output	BINARY(4)	Offset to next project entry. Offset from the beginning of the space to the next authorized project entry. If this value is X'FFFF', the end of the list has been reached.

If there are no authorized projects in the list, one element with blank project name and text is returned, and the next offset is set to X'FFFF'.

Error Messages and Return Codes

The following error messages are sent or returned from QLYALCPR in the error situations described above. Depending on how the ERRCODE parameter has been set, the message data is returned or sent as an *escape* message. In either case, QLYALCPR ends as the result of an error condition. The boldface type in the following messages shows the first-level text, and the unemphasized type shows the second-level text.

ADM1001 Project &1 in use.

Cause: Project &1 is allocated to one or more processes.

Recovery: Try your request again when the project is no longer in use.

ADM1002 Project &1 not found.

Cause: Project &1 does not exist.

Recovery: Use the CRTPRJ command to create a project, or correct the project name. Then try your request again. Use the QRYPRJ command to obtain a list of projects in which you are enrolled, if required.

ADM1012 Project name &1 is not valid.

Cause: The specified project name is not a valid name.

Recovery: Correct the project name and try your request again. The first character in a project name must be alphabetic (A-Z), '\$', '#', or '@'. The remaining characters can be alphanumeric (A-Z, 0-9), '\$', '#', '@', '.', or '_'.

ADM1802 User &1 not enrolled in project &2.

Cause: User &1 is not enrolled in project &2.

Recovery: See the project administrator for project &2.

ADM3002 Action &1 not valid.

Cause: &1 on the action parameter is not valid for the call to QLYALCPR. Valid values for the action parameter are: *CHK, *ALC, *DLC, or *LIST.

Recovery: Specify a valid action parameter value, and try your request again.

The following messages are issued as part of the exception handling for unexpected system errors.

ADM9002 Unexpected error occurred.

Cause: The requested operation could not be performed.

Recovery: See your system administrator to determine the cause of the error.

ADM9003 The requested operation could not be performed.

Recovery: See the detailed messages.

The following user-space error messages are issued using operating-system messages already issued by existing system APIs. These user-space messages are issued as low-level messages and are followed by message ADM9003.

CPF811A User space &4 in &9 damaged.

CPF9801 Object &2 in library &3 not found.

CPF9802 Not authorized to object &2 in &3.

CPF9803 Cannot allocate object &2 in library &3.

CPF9805 Object &2 in library &3 destroyed.

CPF9810 Library &1 not found.

CPF9820 Not authorized to use library &1.

CPF9838 User profile storage limit exceeded.

If an invalid value for the input portion of the ERRCODE parameter was specified (the *bytes provided* field, which should be at least 8 if not zero), the following standard message is sent.

CPF3CF1 Error code parameter not valid.

Examples of How to Use the API

The following examples show how QLYALCPR can be used to check, allocate, or deallocate projects, or to create a list of authorized projects.

Example 1: To check project ABC for authority, existence, and status, type the following:

```
CALL PGM(QLYALCPR) PARM('ABC' *CHK ' ' STATUS ERRCODE)
```

Check the error code and monitor for error messages. If no messages occur, the project is valid and accessible to you.

Example 2: To allocate project ABC, type the following:

```
CALL PGM(QLYALCPR) PARM('ABC' *ALC ' ' STATUS ERRCODE)
```

Check the error code and monitor for error messages. If no messages occur, the project has been allocated successfully.

Example 3: To deallocate project ABC, type the following:

```
CALL PGM(QLYALCPR) PARM('ABC' *DLC ' ' STATUS ERRCODE)
```

Check the error code and monitor for error messages. If no messages occur, the project has been deallocated successfully.

Example 4: To obtain a list of authorized users, use the following C/400 code. Be sure the library QADM is in your library list.

Note: No error conditions are handled in this example.

```

/*****
/*
/* Module Name: API-Test
/*
/* Descriptive Name: Application Development Manager/400 API
/* test program
/*
/* The purpose of this working program is to
/* show users how to use this API. It is meant
/* to make it simple and easy, but users are
/* welcome to change it based on their needs.
/*
/* Language: C/400
/*
*****/

/*****
/* Retrieve various structures/utilities that are used in program.
*****/
#include <stdio.h> /* Standard I/O header
#include <stdlib.h> /* General utilities
#include <string.h> /* String handling utilities
#include <stddef.h> /* Standard definition

/*****
/* Define prototypes of external programs.
*****/
#pragma linkage (QUSCRTUS, OS) /* Linkage to QUSCRTUS -
/* Create User Space API
void QUSCRTUS(char *, char *, int, char *,
char *, char *, char *, char *);
/* Prototype for QUSCRTUS

#pragma linkage (QUSPTRUS, OS) /* Linkage to QUSPTRUS -
/* Rtv ptr to User Spc API
void QUSPTRUS(char *, void *); /* Prototype for QUSPTRUS

#pragma linkage (QUSDLTUS, OS) /* Linkage to QUSDLTUS -
/* Delete User Spacc API
void QUSDLTUS(char *, char *); /* Prototype for QUSDLTUS

#pragma linkage (QLYALCPR, OS) /* Linkage to QLYALCPR -
void QLYALCPR( char *,
char *,
char *,
char *,
char * );

```

```

/*****
/* MAINLINE
*****/
main()
{

char  chUser_Space[] = "PROJLIST QTEMP "; /*User space name,*/
char  chExt_Attr[] = "QTSP "; /* and attribute. */
int   intInitial_Size = 4096; /* Initial user space size. */
char  chInitial_Value[] = "\0"; /* Initial value of the space buf*/
char  chPublic_Auth[] = "*EXCLUDE "; /*Public auth - *EXCLUDE */
char  chText_Desc[50]; /* Text description */
char  chReplace[] = "*YES "; /* Replace existing space */
char  chError_Code[] = "\0"; /* Return error code from System */

/*****
/* Variables used with this API call.
*****/
char  chPrjName[32]; /* Project name. */
char  chAction[10]; /* Action requested. */
char  chUsrSpc[20]; /* User space name/lib. */
char  chStatus[10]; /* Returned status. */
char  chError[30]; /* Returned error. */

/*****
/* Dummy space to hold the list of projects returned by this
/* API. This is done in a simple way without memory alloc
/* so that users may use it in their modules.
*****/
struct AuthProj {
    char  chPrjName[32];
    char  chPrjText[80];
    int   intNxtOff;
};

int  intSize; /* Proj element size. */

struct AuthProj prjList[80]; /* Work buffer to save list info
and users may not need it. */
struct AuthProj *pArrElement = &prjList[0];
struct AuthProj *projLst; /* Pointer to created user space.*/

intSize = sizeof(prjList[0]);

```

```

/*****
/* Create *USRSPC QTEMP/PROJLIST using the system API.          */
*****/
QUSCRTUS(chUser_Space,
         chExt_Attr,
         intInitial_Size,
         chInitial_Value,
         chPublic_Auth,
         chText_Desc,
         chReplace,
         chError_Code);

/*****
/* Setup variables for this API.                                */
*****/
strcpy(chPrjName, " ");
strcpy(chAction, "*LIST    ");

/*****
/* Call this API to build project list.                          */
*****/
QLYALCPR( chPrjName,
         chAction,
         chUser_Space,
         chStatus,
         chError);

/*****
/* Retrieve pointer to user space using the system API.          */
*****/
QUSPTRUS(chUser_Space,
         &projLst);

/*****
/* Loop through project list entries, and copy each entry for   */
/* either print or display.                                     */
*****/
strncpy((void *)pArrElement, (void *)projLst, intSize);

while (prjList[intIndex].intNxtOff != -1) {
    /* Print or display project entry */

    projLst++;                /* Point to next project */
    pArrElement++;           /* Point to next element */
    strncpy((void *)pArrElement, (void *)projLst, intSize);
} /* end while */

/*****
/* Clean up user space.                                          */
*****/
QUSDLTUS(chUser_Space, chError_Code);

}                               /* end of main routine          */

```

Chapter 3. The Program QLYGSPTH

The program QLYGSPTH retrieves the search path for a project, starting at a specific group. The groups in the search path are returned in the *Group_list* parameter. In addition to the names of the groups in the search path, the order of the groups, the promote code, and the AS/400 library name where the group is stored are also returned.

The Call Interface

This program or module, called QLYGSPTH to signify *get search path*, is shipped in the library QADM as part of the Application Development Manager/400 feature. It is built as a user-domain program that is allowed to run in system state. You must have the Application Development Manager/400 feature installed to access this program.

The call has the following format:

```
CALL QLYGSPTH(Project_name,  
              Group_name,  
              Search_path,  
              Group_list,  
              Error_code);
```

Table 4. The parameters for QLYGSPTH

Parameter	Use	Type	Description
Project_name	Input	CHAR(32)	Allowed values—alphanumeric character string
Group_name	Input	CHAR(32)	Allowed values—alphanumeric character string
Search_path	Input	CHAR(10)	Allowed values—alphanumeric character string or *DFT
Group_list	Output	CHAR	This holds the returned list of groups that make up the search path. See “The Group_list parameter” on page 14 for the format of the returned structure.
Error_code	Input/Output	CHAR(*)	Mandatory parameter for the AS/400 APIs

The Project_name parameter: A 32-character string that accepts a valid Application Development Manager/400 project name which is validated by the API. The project name entered is used to identify the project for which you want to obtain the search path.

The Group_name parameter: A 32-character string that accepts a valid Application Development Manager/400 group name which is validated by the API. The group name entered indicates which group starts the search path you are obtaining.

The Search_path parameter: A 10-character string that accepts a valid search path part name or the special value *DFT. The value *DFT indicates that the default search path, starting at the specified group in the specified project, is to be obtained.

The *Group_list* parameter: A structure that is returned. This parameter contains the information about the search path. The following is the format of the returned structure using the C/400 language:

```
typedef struct {
    char groupprj[32];
    char groupname[32];
    int groupdepth;
    int order;
    char groupcode[32];
    char library[10];
    char text[80];
} groupinfo;

struct groups {
    int numgroups;
    groupinfo grou[25];
};
```

The *Error_code* parameter: A mandatory parameter with a structure that is standard to all the AS/400 APIs. This structure is explained in more detail in “The ERRCODE parameter” on page 6.

Error Messages and Return Codes

The following error messages are sent or returned from QLYGSPTH in the error situations described above. Depending on how the *Error_code* parameter has been set, the message data is returned or sent as an *escape* message. In either case, QLYGSPTH ends as the result of an error condition. The boldface type in the following messages shows the first-level text, and the unemphasized type shows the second-level text.

ADM1001 Project &1 is in use.

Cause: Project &1 is allocated to one or more processes.

Recovery: Try your request again when the project is no longer in use.

ADM1002 Project &1 not found.

Cause: Project &1 does not exist.

Recovery: Use the CRTPRJ command to create a project, or correct the project name. Then try your request again. Use the QRYPRJ command to obtain a list of projects in which you are enrolled, if required.

ADM1012 Project name &1 is not valid.

Cause: The specified project name is not a valid name.

Recovery: Correct the project name and try your request again. The first character in a project name must be alphabetic (A-Z), '\$', '#', or '@'. The remaining characters can be alphanumeric (A-Z, 0-9), '\$', '#', '@', '.', or '_'.

ADM1202 Group &1 not found.

Cause: Group &1 was not found in project &2.

Recovery: Use the CRTGRP command to create a group, or correct the group name. Use the PRTPRJ command to obtain a list of all the groups in the project, if required. Then try your request again.

ADM1223 Group name &1 is not valid.

Cause: The specified group name is not a valid name.

Recovery: Correct the group name and try your request again. The first character in a group name must be alphabetic (A-Z), '\$', '#', or '@'. The remaining characters can be alphanumeric (A-Z, 0-9), '\$', '#', '@', '.', or '_'.

ADM1601 Part &1 is in use.

Cause: Part &1 of type &2 in group &3 is in use by another process.

Recovery: Try your request again when the part is not in use.

ADM1615 Part name &1 is not valid.

Cause: The specified part name is not a valid name.

Recovery: Correct the part name and try your request again. The first character in a part name must be alphabetic (A-Z), '\$', '#', or '@'. The remaining characters can be alphanumeric (A-Z, 0-9), '\$', '#', '@', '.', or '_'.

ADM1697 Project must be specified.

Cause: A project name must be specified for this command.

Recovery: Specify an existing project name in the PRJ parameter and try your request again.

ADM1698 Group must be specified.

Cause: A group must be specified for this command.

Recovery: Specify a group name in the GRP parameter and try your request again.

ADM1802 User &1 not enrolled in project &2.

Recovery: See the project administrator or QSECOFR for project &2.

ADM4708 Too many groups in the search path.

Cause: There are too many groups in the search path to add to the library list. A maximum of 25 groups can be listed in a library list.

Recovery: Reduce the number of search path groups or remove some libraries from the library list. Then try your request again.

ADM4801 Search path part not found.

Recovery: See the detailed messages.

ADM4802 SCHPTH part &1 not found.

Cause: Part &1 of type SCHPTH could not be found.

Recovery: Correct the name of the SCHPTH part and try your request again.

ADM4804 Part &4 could not be processed.

Cause: An error occurred when processing part &4 of type &3 in group &2 of project &1.

Recovery: See the detailed messages.

ADM4805 SCHPTH part &4 contains no information.

Cause: Part &4 of type &3 in group &2 of project &1 contains no information.

Recovery: Specify another part of type SCHPTH or put the search path information in part &4. Then try your request again.

ADM4806 SCHPTH part &4 contains bad record.

Cause: Part &4 of type &3 in group &2 of project &1 contains a record with a search path format that is not valid. Each record in a part of type SCHPTH must contain a project name followed by a group name with the two fields separated by one or more blanks.

Recovery: Correct the incorrect record in part &4 of type SCHPTH.

ADM4807 SCHPTH part &4 could not be verified.

Cause: Part &4 of type &3 in group &2 of project &1 contains a search path that could not be verified.

Recovery: See the detailed messages.

ADM4808 The record length for &4 is too long.

Cause: Part &4 of type &3 in group &2 of project &1 has a record length greater than the maximum of 240 characters.

Recovery: Use the DLTPART command to delete part &4 and create a new part of type SCHPTH using the CRTPART command. Then try your request again.

ADM4809 Default search path could not be retrieved.

Recovery: See the detailed messages.

ADM4810 Error occurred when processing search path.

Recovery: See the detailed messages.

ADM5009 Project &1 is in use or needs to be reclaimed.

Cause: Project &1 is in use or needs to be reclaimed because it is in an inconsistent state.

Recovery: Try your request again. If you continue to receive this message, ask your project administrator or QSECOFR to reclaim project &1 by running the RCLPRJ command.

The following messages are issued as part of the exception handling for unexpected system errors:

ADM9002 Unexpected error occurred.

Cause: The requested operation could not be performed.

Recovery: See your system administrator to determine the cause of the error.

ADM9003 The requested operation could not be performed.

Recovery: See the detailed messages.

If an invalid value for the input portion of the *Error_code* parameter was specified (the *bytes provided* field, which should be at least 8 if not zero), the following standard message is sent:

CPF3CF1 Error code parameter not valid.

Examples of How to Use the API

The following examples show how QLYGSPTH can be used to obtain the search path information for a group:

Example 1: To obtain the default search path starting from the group DEV1, project ABC, type the following:

```
CALL PGM(QLYGSPTH) PARM('ABC' 'DEV1' '*DFT' Group_list Error_code)
```

Check the error code and monitor for error messages. If no messages occur, the search path is accessible to you in the returned *Group_list* structure.

Example 2: To obtain the VERSION_2 search path starting from the group DEV1, project ABC, type the following:

```
CALL PGM(QLYGSPTH) PARM('ABC' 'DEV1' 'VERSION_2'
Group_list Error_code)
```

Check the error code and monitor for error messages. If no messages occur, the search path is accessible to you in the returned *Group_list* structure.

Chapter 4. The Program QLYVUDPT

The program QLYVUDPT determines if a part type is a user-defined part type, and if it is, whether the user-defined part type is stored in a source file member.

The Call Interface

This program or module, called QLYVUDPT to signify *validate user-defined part type*, is shipped in the library QADM as part of the Application Development Manager/400 feature. It is built as a user-domain program that is allowed to run in system state. You must have the Application Development Manager/400 feature installed to access this program.

The call has the following format:

```
CALL QLYVUDPT(Part_type,  
              UDT_flag,  
              Member_flag,  
              Error_code);
```

Table 5. The parameters for QLYVUDPT

Parameter	Use	Type	Description
Part_type	Input	CHAR(10)	Allowed values—alphanumeric character string
UDT_flag	Output	CHAR(1)	Allowed values—0, 1
Member_flag	Output	CHAR(1)	Allowed values—0, 1
Error_code	Input/Output	CHAR(*)	Mandatory parameter for the AS/400 APIs

The Part_type parameter: A 10-character string. This is the name of the part type that you want to identify as a user-defined part type.

The UDT_flag parameter: Returns a value of 0 or 1. The value 0 indicates that the type entered on the *Part_type* parameter is not a user-defined part type. The value 1 indicates that the type entered on the *Part_type* parameter is a user-defined part type.

The Member_flag parameter: Returns a value of 0 or 1. The value 0 indicates that the type entered on the *Part_type* parameter is not stored in a source file member. The value 1 indicates that the type entered on the *Part_type* parameter is stored in a source file member. This value is always returned if the part type is not user-defined.

The Error_code parameter: A mandatory parameter with a structure that is standard to all the AS/400 APIs. This structure is explained in more detail in “The ERRCODE parameter” on page 6.

Error Messages and Return Codes

The following error messages are sent or returned from QLYVUDPT in the error situations described above. Depending on how the *Error_code* parameter has been set, the message data is returned or sent as an *escape* message. In either case, QLYVUDPT ends as the result of an error condition.

ADM9002 Unexpected error occurred.

Cause: The requested operation could not be performed.

Recovery: See your system administrator to determine the cause of the error.

ADM9003 The requested operation could not be performed.

Recovery: See the detailed messages.

If an invalid value for the input portion of the *Error_code* parameter was specified (the *bytes provided* field, which should be at least 8 if not zero), the following standard message is sent.

CPF3CF1 Error code parameter not valid.

Example of How to Use the API

The following example shows how QLYVUDPT can be used to determine if a part type is user-defined, and if it is, whether the part type is stored in a source file member.

Example: To determine whether the part type PL1 is a user-defined part type, type the following:

```
CALL PGM(QLYVUDPT) PARM('PL1' UDT_flag Member_flag Error_code)
```

Check the error code and monitor for error messages. If no messages occur, the values in *UDT_flag* and *member_flag* are accessible to you.

Chapter 5. The External Data Files

The Application Development Manager/400 external data files refer to the logical views of certain files in the Application Development Manager/400 database. In a general sense, they provide the information about hierarchies, projects, groups, and parts. The Application Development Manager/400 feature stores this information in the library QUSRSYS. This information is contained in the restricted data store and involves certain AS/400 objects. There is only one set of these files, and each set contains the information for all projects.

The files QALYPROJS, QALYGROUPS, QALYPARTS, and QALYPART1 are described to Application Development Manager/400 users. They are shipped with public authority *USE. Developers can read these files if they are authorized to them.

The Project Registration File

QALYPROJS is a logical file that lists the projects on the system. You can use this file to obtain a list of projects in much the same way the programming development manager utility obtains a list.

When an administrator creates a project, that project is *registered* to the Application Development Manager/400 database, and a record is created in the file QALYPROJS. There is one record in this file for each project on the system.

A	R PROJREGS		TEXT('PROJECT REGISTRATION') +
A			PFILE(QADM/QALYPRJLST)
A	PROJECT	32A	COLHDG('PROJECT NAME')
A	IGCDATA	1A	COLHDG('IGC DATA')
/*STDA	DESCR	80A	COLHDG('PROJECT DESCRIPTION')
/*IGCA	DESCR	800	COLHDG('PROJECT DESCRIPTION')
A	LSTCHG	14A	COLHDG('LAST CHANGE')
A	K PROJECT		

Figure 1. The project registration file QALYPROJS

The information registered with each group is provided in the following list. The (K) symbol denotes the key fields in the file.

- (K)Project name—Char(32). The name of the project.
Project name is defined to have a unique value. This means that there cannot be two projects with the same name.
- IGC data—Char(1). 1 = project can contain IGC data. 0 = project cannot contain IGC data.
- Project description—Char(80).
Note: When the Application Development Manager/400 feature is installed, this field is defined to have the coded character set identifier (CCSID) of the base operating system.
- Last change date and time—the format is YYYYMMDDHHMMSS. The date and time that the project last changed.

The Group Hierarchy File

QALYGROUPS is a logical file that lists the groups in the projects on the system. You can use this file to obtain a list of groups in much the same way the programming development manager utility obtains a list.

When an administrator defines a project's hierarchy, that information is registered to the Application Development Manager/400 database, and a record is created in the file QALYGROUPS. There is one record in this file for each group in all the projects on the system.

A	R	PRJGRPS		TEXT('PROJECT GROUPS') +
A				PFILE(QADM/QALYPRJD)
A		PROJECT	32A	COLHDG('PROJECT NAME')
A		GROUP	32A	COLHDG('GROUP NAME')
A		GRPORD	3S 0	COLHDG('GROUP ORDER')
A		SGROUP	5A	COLHDG('SHORT GROUP NAME')
A		LIBRARY	10A	COLHDG('LIBRARY NAME')
A		PARENTGRP	32A	COLHDG('PARENT GROUP')
A		GRPDEPTH	3S 0	COLHDG('GROUP DEPTH')
A		PCODE	32A	COLHDG('PROMOTE CODE')
A		CCSID	2A	COLHDG('CCSID')
/*STDA		DESCR	80A	COLHDG('GROUP DESCRIPTION')
/*IGCA		DESCR	800	COLHDG('GROUP DESCRIPTION')
A	K	PROJECT		
A	K	GRPORD		

Figure 2. The group hierarchy file QALYGROUPS

The information registered with each group is provided in the following list. The (K) symbol denotes the key fields in the file.

- (K)Project name—Char(32). The name of the project.
- Group name—Char(32). The name of the group.
- (K)Group order—Zoned (3,0). The depth-first search order of the groups in the project hierarchy. This search order goes down the first branch in a hierarchy to the bottom, then down the second branch, and so on.

Note: The fields Project name and Group name are defined to have unique values. This means that a project cannot contain two groups with the same name.

- Short group name—Char(5). The short name of the group.
- Library name—Char(10). The name of the library associated with the group.
- Parent group—Char(32). The name of the parent group. *NONE denotes the root group.
- Group depth—Zoned (2,0). The depth of the group in the group hierarchy. 01 = Root, 02 = Child of root, 03 = Grandchild of root, and so on. The limit of a depth of 99 groups.
- Promote code—Char(32). The valid promote code. *NONE shows that no promote code was defined for the group.
- CCSID—Char(2). The coded character set identifier of the group.

- Group description—Char(80). The description for the group.

Note: When the Application Development Manager/400 feature is installed, this field is defined to have the CCSID of the base operating system.

The Part Directory Files

QALYPARTS is a logical file that lists the parts in all the groups in all projects on the system. You can use this file to obtain a list of parts in much the same way as the programming development manager utility obtains a list.

A	R PARTDIRS		TEXT('PART DIRECTORY DATA ST+
A			ORE') PFILE(QADM/QALYPART)
A	PROJECT	32A	COLHDG('PROJECT NAME')
A	TYPE2	4S 0	COLHDG('INTERNAL PART TYPE')
A	PART	32A	COLHDG('PART NAME')
A	GROUP	32A	COLHDG('GROUP NAME')
A	TYPE	32A	COLHDG('PART TYPE')
A	LANGUAGE	10A	COLHDG('PART LANGUAGE')
A	LIBRARY	10A	COLHDG('NATIVE LIBRARY')
A	SRCFILE	10A	COLHDG('PART SOURCE FILE')
A	LGLDATE	8A	COLHDG('LOGICAL DATE')
A	LGLTIME	6A	COLHDG('LOGICAL TIME')
A	CRTDATE	8A	COLHDG('CREATE DATE')
A	CRTTIME	6A	COLHDG('CREATE TIME')
A	PCODE	32A	COLHDG('PROMOTE CODE')
A	LCUSER	10A	COLHDG('LAST CHANGED USER')
A	OBJTYPE	10A	COLHDG('OBJECT TYPE')
A	OBJATTR	10A	COLHDG('OBJECT ATTRIBUTE')
/*STDA	DESC	80A	COLHDG('PART DESCRIPTION')
/*IGCA	DESC	800	COLHDG('PART DESCRIPTION')
A	PLCHLD	1A	COLHDG('RESERVED')
A	K PROJECT		
A	K TYPE2		
A	K PART		
A	K GROUP		
A	0 PLCHLD		COMP(EQ '1')

Figure 3. The part directory file QALYPARTS

The directory of parts in a project can be stored in the file QALYPARTS. The information registered with each group is provided in the following list. The **(K)** symbol denotes the key fields in the file.

- **(K)**Project name—Char(32). The name of the project.
- **(K)**Internal part type—Zoned(4,0). An internal number associated with the type of the part. The fields are ordered the same way as displayed by the Programming Development Manager utility.
- **(K)**Part name—Char(32). The name of the part. This field is 32 characters long for future growth.
- **(K)**Group name—Char(32). The name of the group.

Note: The key fields Project name, Internal part type, Part name, and Group name are defined to have unique values. This means that there cannot be two parts with the same name.

- Part type—Char(32). Type of the part. This field is 32 characters long for future growth.
- Part language—Char(10).
- Native library—Char(10). The AS/400 library containing the part.
- Part source file (for xxxSRC, xxxINC, BLDOPT, and SCHPTH)—Char(10). The name of the source file that stores the member associated with the part.
- Logical date—YYYYMMDD. Last changed date.
- Logical time—HHMMSS. Last changed time.
- Create date—YYYYMMDD.
- Create time—HHMMSS.
- Promote code—Char(32).
- Last changed user—Char(10).
- Object type—Char(10). Type of the object. If the object is a source member, this is the member type.
- Object attribute—Char(10). Attribute of the associated object. If the object is a source member, this field is blank.
- Part description—Char(80). The description for the part.
Note: When the Application Development Manager/400 feature is installed, this field is defined to have the CCSID of the base operating system.
- Reserved —Char(1). Y/N. This is for internal use only by the Application Development Manager/400 feature.

The logical file QALYPART1 also lists the parts in all the groups in all projects on the system. The only differences between this file and QALYPARTS are:

- This file sorts user-defined types in alphabetic order *after* it sorts system-supplied types.
- It includes information about the access key.

A			UNIQUE
A	R PARTDIRS		TEXT('PART DIRECTORY DATA STORE') PFILE(QADM/QALYPART)
A			COLHDG('PROJECT NAME')
A	PROJECT	32A	COLHDG('INTERNAL PART TYPE')
A	TYPE2	4S 0	COLHDG('PART TYPE')
A	TYPE	32A	COLHDG('PART NAME')
A	PART	32A	COLHDG('GROUP NAME')
A	GROUP	32A	COLHDG('PART LANGUAGE')
A	LANGUAGE	10A	COLHDG('NATIVE LIBRARY')
A	LIBRARY	10A	COLHDG('PART SOURCE FILE')
A	SRCFILE	10A	COLHDG('LOGICAL DATE')
A	LGLDATE	8A	COLHDG('LOGICAL TIME')
A	LGLTIME	6A	COLHDG('CREATE DATE')
A	CRTDATE	8A	COLHDG('CREATE TIME')
A	CRTTIME	6A	COLHDG('PROMOTE CODE')
A	PCODE	32A	COLHDG('LAST CHANGED USER')
A	LCUSER	10A	COLHDG('OBJECT TYPE')
A	OBJTYPE	10A	COLHDG('OBJECT ATTRIBUTE')
A	OBJATTR	10A	COLHDG('PART HOLDER USERID')
A	ACCESSKEY	10A	COLHDG('PART DESCRIPTION')
/*STDA	DESC	80A	COLHDG('PART DESCRIPTION')
/*IGCA	DESC	800	COLHDG('RESERVED')
A	PLCHLD	1A	
A	K PROJECT		
A	K TYPE2		
A	K TYPE		
A	K PART		
A	K GROUP		
A	O PLCHLD		COMP(EQ '1')

Figure 4. The part directory file QALYPART1

The directory of parts in a project can also be stored in the file QALYPART1. The information registered with each group is provided in the following list. The **(K)** symbol denotes the key fields in the file.

- **(K)**Project name—Char(32). The name of the project.
- **(K)**Internal part type—Zoned(4,0). An internal number associated with the type of the part. The fields are ordered the same way as displayed by the Programming Development Manager utility.
- **(K)**Part type—Char(32). The type of the part.
- **(K)**Part name—Char(32). The name of the part. This field is 32 characters long for future growth.
- **(K)**Group name—Char(32). The name of the group. This field is 32 characters long for future growth.

Note: The key fields Project name, Internal part type, Part type, Part name, and Group name are defined to have unique values. This means that there cannot be two parts with the same name.

- Part language—Char(10).
- Native library—Char(10). The AS/400 library containing the part.
- Part source file (for xxxSRC, xxxINC, BLDOPT, and SCHPTH)—Char(10). The name of the source file that stores the member associated with the part.
- Logical date—YYYYMMDD. Last changed date.
- Logical time—HHMMSS. Last changed time.
- Create date—YYYYMMDD.
- Create time—HHMMSS.
- Promote code—Char(32).
- Last changed user—Char(10).
- Object type—Char(10). Type of the object. If the object is a source member, this is the member type.
- Object attribute—Char(10). Attribute of the associated object. If the object is a source member, this field is blank.
- Access key—Char(10). Identifies the user profile of the developer who has checked the part out.
- Part description—Char(80). The description for the part.
Note: When the Application Development Manager/400 feature is installed, this field is defined to have the CCSID of the base operating system.
- Reserved —Char(1). Y/N. This is for internal use only by the Application Development Manager/400 feature.

QALYPART2 is a logical file that lists the parts in all the groups in all projects on the system. You can use this file to obtain a list of Application Development Manager/400 four part names using the PART (source member name), SRCFILE (source file name), and LIBRARY. This file is useful for parts stored in source members.

A	R PARTDIR2		TEXT('PART DIRECTORY DATA ST+ ORE') PFILE(QADM/QALYPART)
A			COLHDG('PROJECT NAME')
A	PROJECT	32A	COLHDG('GROUP NAME')
A	GROUP	32A	COLHDG('INTERNAL PART TYPE')
A	TYPE2	4S 0	COLHDG('PART TYPE')
A	TYPE	32A	COLHDG('PART NAME')
A	PART	32A	COLHDG('NATIVE LIBRARY')
A	LIBRARY	10A	COLHDG('PART SOURCE FILE')
A	SRCFILE	10A	
A	K PART		
A	K LIBRARY		
A	K SRCFILE		

Figure 5. The part directory file QALYPART2

The directory of parts in a project can be stored in the file QALYPART2. The information registered with each group is provided in the following list. The **(K)** symbol denotes the key fields in the file.

Note: In case of objects, such as *PGM and *DTAARA, it is possible that multiple records matching a single set of key field values may be found in this part directory file. You can use the values returned from this file to get a record from QALYPARTS or QALYPART1 part directory files and then match the object attribute to find the unique four part name.

- Project name—Char(32). The name of the project.
- Group name—Char(32). The name of the group.
- Internal part type—Zoned(4,0). An internal number associated with the type of the part. The fields are ordered the same way as displayed by the Programming Development Manager utility.
- Part type—Char(32). Type of the part. This field is 32 characters long for future growth.
- **(K)**Part name—Char(32). The name of the part. This field is 32 characters long for future growth.
- **(K)**Native library—Char(10). The AS/400 library containing the part source file.
- **(K)**Part source file (for xxxSRC, xxxINC, BLDOPT, and SCHPTH)—Char(10). The name of the source file that stores the member associated with the part.

Note: The value of this field is blank in case of objects, such as *PGM and *DTAARA.

Index

A

allocating a project 3
authority checking 3

D

deallocating a project 3

E

examples

- using QLYALCPR program 8
- using QLYGSPTH program 17
- using QLYVUDPT program 20

existence checking 3

external data files

- description 21
- QALYGROUPS 22
- QALYPART1 25
- QALYPART2 26
- QALYPARTS 23
- QALYPROJS 21

G

group hierarchy file, QALYGROUPS 22

L

listing authorized users 3

P

part directory file, QALYPART1 25

part directory file, QALYPART2 26

part directory file, QALYPARTS 23

program, QLYALCPR 3

program, QLYGSPTH 13

program, QLYVUDPT 19

programs

- QLYALCPR 3
- QLYGSPTH 13
- QLYVUDPT 19

project registration file, QALYPROJS 21

Q

QALYGROUPS, description 22

QALYGROUPS, group hierarchy file 22

QALYPART1, description 25

QALYPART1, part directory file 25

QALYPART2, description 26

QALYPART2, part directory file 26

QALYPARTS, description 23

QALYPARTS, part directory file 23

QALYPROJS, description 21

QALYPROJS, project registration file 21

QLYALCPR

- ACTION parameter, description 4

- description of parameters 4

- ERRCDE parameter, description 6

- format of CALL command 4

- project list structure 6

- PROJECT parameter, description 4

- STATUS parameter, description 5

- using examples 8

- USRSPC parameter, description 5

QLYGSPTH

- description of parameters 13

- description of program 13

- ERROR_CODE parameter, description 14

- examples 17

- format of CALL command 13

- GROUP_LIST parameter, description 14

- GROUP_NAME parameter, description 13

- PROJECT_NAME parameter, description 13

- SEARCH_PATH parameter, description 13

QLYVUDPT

- description of parameters 19

- description of program 19

- ERROR_CODE parameter, description 19

- examples 20

- format of CALL command 19

- MEMBER_FLAG parameter, description 19

- PART_TYPE parameter, description 19

- UDT_FLAG parameter, description 19

QUSRSYS library 21

R

retrieving the search path for a project 13

S

sample files

- QALYGROUPS 22

- QALYPART1 25

- QALYPART2 26

- QALYPARTS 23

- QALYPROJS 21

status checking 3

U

using QLYALCPR program	8
using QLYGSPTH program	17
using QLYVUDPT program	20