Software Product APIs (V5R2)

Table of Contents

- Software Product APIs
- APIs
  - Accept Software Agreement (QLPACAGR)
  - Add License Key Information (QLZAADDK)
  - Add or Remove Product Support (QSZSPTPR)
  - Add Product License Information (QLZADDLI)
  - Check Target Release (QSZCHKTG)
  - Copy Program Temporary Fix to Save File (QPZCPYSV)
  - Create Product Definition (QSZCRTPD)
  - Create Product Load (QSZCRTPL)
  - Create Program Temporary Fix (QPZCRTFX)
  - Create PTF Group (QpzCreatePtfGroup)
  - Delete Product Definition (QSZDLTPD)
  - Delete Product Load (QSZDLTPL)
  - Delete PTF Group (QpzDeletePtfGroup)
  - Delete Registered Application Files (QSZDLTAF, QszDltRegAppFiles)
  - Generate CD-ROM Premastering Information (QLPCDINF, QlpGenCdPremasteringInfo)
  - Generate License Key (QLZAGENK)
  - Generate Program Temporary Fix Name (QPZGENNM)
  - Handle CD-ROM Premastering State (QLPCDRST, QlpHandleCdState)
  - Install Secondary Language (QLPISLNG)
  - List Product in a Save File (QLPLPRDS)
  - List Program Temporary Fixes (QpzListPTF)
  - List PTF Group Details (QpzListPtfGroupDetails)
  - List PTF Groups (QpzListPtfGroups)
  - List Registered Application Information (QSZLSTRA, QszListRegAppInfo)
  - Log Program Temporary Fix Information (QPZLOGFX)
  - Package Product Option (QSZPKGPO)
  - Release License (QLZARLS)
  - Request License (QLZAREQ)

    - Request Order Assistance (QMARQSOA)
    - Retrieve License Information (QLZARTV)
    - Retrieve License Key Information (QLZARTVK)
    - Retrieve Product Information (QSZRTVPR)
    - Retrieve Program Temporary Fix Information (QPZRTVFX)
    - Select Product (QSZSLTPR)
    - Update iSeries Registered Application Information Repository (QSZUPDRA, QszUpdRegAppInfoRepository)
- Exit Programs
    - Program Temporary Fix
    - QLPUSER
    - Software Product Functions

Software Product APIs

Software product APIs let you work with software products and program temporary fixes (PTFs) on your system. With these APIs, you can:

- Work with the software license management for a product
- Create and delete product definition and product load objects
- Package one or more product loads for a specified product option
- Retrieve product information about a specific product load
- Create PTFs, retrieve PTF information, and log PTF information
- Retrieve a list of products

You may write exit programs that are called by programs and by program temporary fixes (PTFs).

For information about packaging and managing software products, see the System Manager Use book. The steps for packaging a product are the same whether you use the System Manager licensed program commands or the APIs in this section.

The software product APIs are:

- »Accept Software Agreement (QLPACAGR) records the acceptance of the software agreement for a product.«
- Add License Key Information (QLZAADDK) allows you to add license key information to the license repository.
- Add or Remove Product Support (QSZSPTPR) adds or removes support to a product.
- Add Product License Information (QLZADDLI) adds license information to a product or feature.
- Check Target Release (QSZCHKTG) verifies that a valid target release value is specified on a CL command that supports the TGTRLS parameter.
- Copy Program Temporary Fix to Save File (QPZCPYSV) allows you to copy PTFs for the selected product from the media and store them in *SERVICE.
- Create Product Definition (QSZCRTPD) creates a product definition object.
- Create Product Load (QSZCRTPL) creates a product load object.
- Create Program Temporary Fix (QPZCRTFX) creates a PTF save file and optionally creates a PTF cover letter.
- »Create PTF Group (QpzCreatePtfGroup) creates a PTF group.«
- Delete Product Definition (QSZDLTPD) deletes a product definition object.
- Delete Product Load (QSZDLTPL) deletes a single product load object.
- »Delete PTF Group (QpzDeletePtfGroup) deletes a PTF group from the system.«
- Delete Registered Application Files (QSZDLTAF, QszDltRegAppFiles) deletes the files listed in the files tag for the given component.
- Generate CD-ROM Premastering Information (QLPCDINF, QlpGenCdPremasteringInfo) generates the distribution set map file. This API also retrieves information about the files that were saved when the job was enabled for CD-ROM premastering using the Handle CD-ROM Premastering State (QLPCDRST, QlpHandleCdState) API.

- Generate License Key (QLZAGENK) generates a license key to enable users to access a product or a feature of a product.
- Generate Program Temporary Fix Name (QPZGENNM) generates a unique name for PTF save files and cover letters.
- Handle CD-ROM Premastering State (QLPCDRST, QlpHandleCdState) enables and disables the job for CD-ROM premastering. This API also queries the current CD-ROM premastering job state.
- Install Secondary Language (QLPISLNG) installs the secondary language that is specified during interactive mode or batch mode of installation.
- List Product in a Save File (QLPLPRDS) generates a list containing product ID, release level, option, load type, and language ID for all product loads found in a save file.
- List Program Temporary Fixes (QpzListPTF) returns a list of PTFs for the specified product, option, load, and release.
- »List PTF Group Details (QpzListPtfGroupDetails) lists information for a specific PTF group on the system.«
- »List PTF Groups (QpzListPtfGroups) returns a list of all PTF groups that are known to the system.«
- List Registered Application Information (QSZLSTRA, QszListRegAppInfo) retrieves the results of a query of the iSeries Registered Application Information Repository.
- Log Program Temporary Fix Information (QPZLOGFX) logs that a PTF has been received on the system and can be displayed or loaded from device *SERVICE.
- Package Product Option (QSZPKGPO) packages one or more product loads for a specified product option.
- Release License (QLZARLS) releases a use of the license for the product.
- Request License (QLZAREQ) requests a use of the license for the product.
- Request Order Assistance (QMARQSOA) sends a request for order assistance to a service provider using an ECS connection, and creates a corresponding entry in the user''s order log.
- Retrieve License Information (QLZARTV) retrieves the license information for a software product.
- Retrieve License Key Information (QLZARTVK) retrieves the license key information for the specified product, license terms, and features for the specified systems from the license repository.
- Retrieve Product Information (QSZRTVPR) retrieves information about a specific product load for a software product.
- Retrieve Program Temporary Fix Information (QPZRTVFX) returns information about a specific program temporary fix (PTF).
- Select Product (QSZSLTPR) displays or retrieves a list of products.
- Update OS/400 Registered Application Information Repository (QSZUPDRA, QszUpdRegAppInfoRepository) updates information about one or many separately installable pieces of an application-called component.

The software product exit programs are:

- Program Temporary Fix is called when a PTF is temporarily or permanently applied or removed with the Apply PTF (APYPTF) or Remove PTF (RMVPTF) commands.
- QLPUSER is called during the automatic installation process and can be used by central sites when they are distributing products to remote locations.
- Software Product Functions are specified when creating products that will be restored, deleted, saved,

and checked with CL commands.

Accept Software Agreement (QLPACAGR) API

Required Parameter Group:

1	Product ID	Input	Char(7)
2	Product release	Input	Char(6)
3	Product option	Input	Char(4)
4	Error code	I/O	Char(*)

Default Public Authority: *USE

Threadsafe: No

The Accept Software Agreement (QLPACAGR) API records the acceptance of the software agreement for a product. It is assumed that the caller of this API has previously displayed and obtained acceptance for the terms of the agreement.

Authorities and Locks

Public API authority

*USE authority to RSTLICPGM command.

Required Parameter Group

Product ID

INPUT; CHAR(7)

The 7-character ID of the product for which the software agreement is being accepted. The product ID must be in the format nxxxxxx, where n is any numeric character 0 through 9, and x is any numeric character 0 through 9 or uppercase letter A through Z.

Product release

INPUT; CHAR(6)

The version, release, and modification level of the product for which the software agreement is being accepted. The release must be in the format VxRyMz. Valid values for x and y are 0 through 9. Valid values for z are 0 through 9 or A through Z. For example, V5R2M0 is version 5, release 2, modification 0.

Product option

INPUT; CHAR(4)

The option number of the product for which the software agreement is being accepted. Use 0000 for the base option. Valid values are 0000 through 0099, where each character is a digit.

Error code

I/O; CHAR(*)

The structure in which to return error information. For the format of the structure, see Error code parameter.

Error Messages

Message ID	Error Message Text
CPF358A E	Release not valid.
CPF3DDD E	Unable to accept software agreement.
CPF3DDE E	Product identifier &1 option &2 not allowed.
CPF3DDF E	Product option &2 not valid.
CPF3DEF E	Product identifier &1 not valid.
CPF3DF3 E	Not authorized to accept software agreement.
CPF24B4 E	Severe error while addressing parameter list.
CPF3CF1 E	Error code parameter not valid.
CPF9872 E	Program or service program &1 in library &2 ended. Reason code &3.

API Introduced: V5R2

Add License Key Information (QLZAADDK) API

Required Parameter Group:

1	License key information	Input	Char(*)
2	License key information format name	Input	Char(8)
3	Number of license key records added	Output	Binary(4)
4	Error code	I/O	Char(*)

Default Public Authority: *EXCLUDE

Threadsafe: No

The Add License Key Information (QLZAADDK) API allows you to add license key information to the license repository. The license repository is used to store license key information. It contains a record for each license for every unique product, license term, feature, and system. The repository may contain licenses for any system, and the product does not need to be installed.

If a key already exists in the repository for the given product, license term, feature, and system, it is replaced. If the product is installed on the system and the license is for that system, the license is also installed. That is, the usage limit is changed from the product's default usage limit to the licensed usage limit. The expiration date is also set based on the license key.

If this is the first license key added for this product, the threshold is set to 90 percent of the usage limit. Message queues default to the message queues defined in the licensed information for the operating system. Violations are not logged. If a license was already installed on the system, these values are not changed. To change any of these default values, you must use the Change License Information (CHGLICINF) command.

Authorities and Locks

API QLZAADDK Authority

*PUBLIC(*EXCLUDE)

Required Parameter Group

License key information

INPUT; CHAR(*)

The license key information that is to be added to the repository. The structure of this information is determined by the name of the format.

License key information format name

INPUT; CHAR(8)

The name of the format containing the license key information.

The format name is:

LICA0100 The license key information to be added to the repository. For details, see the LICA0100 Format.

Number of license key records added

OUTPUT; BINARY(4)

The number of license key records that were successfully added to the repository.

Error code

I/O; CHAR(*)

The structure in which to return error information. For the format of the structure, see Error Code Parameter.

LICA0100 Format

The following is the license information which is to be added to the repository. For detailed descriptions of the fields, see Field Descriptions.

Offset		Type	Field
Dec	Hex		
0	0	BINARY(4)	Size of structure
4	4	BINARY(4)	Offset to the first license key record
8	8	BINARY(4)	Number of license key records in list
12	C	BINARY(4)	Size of license key record
16	10	BINARY(4)	Reserved
Note: The offsets of the following fields vary. These fields repeat, in the order listed, for each license key record.			
		CHAR(7)	Product ID
		CHAR(6)	License term
		CHAR(4)	Feature
		CHAR(8)	System serial number
		CHAR(4)	Processor group
		CHAR(3)	Reserved
		BINARY(4)	Usage limit
		CHAR(7)	Expiration date
		CHAR(8)	Vendor data
		CHAR(18)	License key
		CHAR(15)	Reserved
		CHAR(*)	Reserved

Field Descriptions

Expiration date. The date the license will expire.

The valid values are:

CYYMMDD C is the century, YY is the year, MM is the month, and DD is the day. Century, where 0 indicates years 19*xx* and 1 indicates years 20*xx*.

- Month may not be greater than 12.
- Day may not be greater than 31.

The date must be numeric as follows:

9999999 The license does not have an expiration date.

Feature. The feature of the product. Valid values for the feature are 5001 through 9999.

License key. The license key provided by the vendor. The license key contains only the letters A - F and the numbers 0 - 9.

License term. The extent of time the authorized usage limit for a product lasts. Each time a new license term is installed for a product, the authorized usage limit must be set by:

- Obtaining a new license key.
- Using the Add License Key Information (ADDLICKEY) command.

Possible values are:

Vx The authorized usage limit is valid for the entire version of the product or feature.

VxRy The authorized usage limit is valid only for the entire release of the product or feature.

VxRyMz The authorized usage limit is valid only for the modification level of the product.

Where the x and y can be a number from 0 through 9. Z can be a number 0 through 9 or a letter A through Z.

Number of license key records in list. The number of license key records in the list.

Offset to the first license key record. The byte offset from the beginning of the license key information variable to the start of the license key information records.

Processor group. The processor group that this license is for. This field is left justified.

The valid special value is:

**ANY* The license key is valid for any processor group.

Product ID. The product ID of the product or feature for which license key information is to be added to the repository.

Reserved. If this field is input, character fields must be set to blanks and binary fields must be set to hexadecimal zeros.

Size of license key record. The size of each license key record. This field can be used to get the offset of the next license key record.

Size of structure. The size of the entire data passed in on this parameter.

System serial number. The system serial number for which license key information is added to the repository.

The valid special value is:

**LOCAL* License key information for only this local system will be added to the repository.

The special values are left justified. A value other than a special value must be right justified.

Usage limit. The usage limit for this license.

-1 There is no maximum number of license users for this product.

0-999999 The maximum number of license users for this product.

Vendor data. The data for this field, along with the license key information, comes from the software provider.

Error Messages

Message ID	Error Message Text
CPD9E2D E	Usage limit cannot be less than current usage.
CPF24B4 E	Severe error while addressing parameter list.
CPF3C21 E	Format name &1 is not valid.
CPF3C90 E	Literal value cannot be changed.
CPF3CF1 E	Error code parameter not valid.
CPF9822 E	Not authorized to file &1 in library &2.
CPF9826 E	Cannot allocate file &2.
CPF9872 E	Program or service program &1 in library &2 ended. Reason code &3.
CPF9E15 E	Error in license management function.
CPF9E54 E	License term &1 not valid.
CPF9E59 E	Expiration date &1 is not valid.
CPF9E5A E	Usage limit not valid.
CPF9E6C E	The license key cannot be used for processor group &2.
CPF9E68 E	Product &1 license term &2 feature &3 not found.
CPF9E6D E	Feature &3 not valid.

CPF9E6E E	Product identifier &1 not valid.
CPF9E74 E	License key not valid.
CPF9E81 E	Cannot install keys for product &1.

API Introduced: V3R1

Top | Software Product APIs | APIs by category

Add or Remove Product Support (QSZSPTPR) API

Required Parameter Group:

1	Product information	Input	Char(*)
2	Length of product information	Input	Binary(4)
3	Product information format name	Input	Char(8)
4	Requested action	Input	Binary(4)
5	Error Code	I/O	Char(*)

Default Public Authority: *USE

Threadsafe: No

The Add or Remove Product Support (QSZSPTPR) API adds or removes support to a product. The action is performed by specifying a product ID, release, option number, and load ID. When you are adding support to a product, you indicate that you want to be able to order and receive program temporary fixes (PTFs) to maintain that product. An information entry is also created for that product.

Removing product support does not remove related PTF save files or PTF information. PTFs for products that are not supported, however, cannot be distributed using the System Manager/400 licensed program.

Authorities and Locks

Product Availability Authority

None

Product Availability Lock

*SHRRD. The product availability object resides in the QUSRSYS library.

Product Definition Authority

None

Product Load Authority

None

Required Parameter Group

Product information

INPUT; CHAR(*)

The product for which support is to be added or removed. For the structure of this parameter, see Format of Product Information Parameter.

Length of product information

INPUT; BINARY(4)

The length of the product information parameter in bytes. The value specified must be at least 53.

Product information format name

INPUT; CHAR(8)

The content and format of the product information. The possible format name is:

SPTP0100 The product information needed to add or remove support.

Requested action

INPUT; BINARY(4)

The action to be performed against each of the products in the list. Valid values are:

1 Adds support to the product passed in the first parameter. When you use this value, you indicate that you want to provide service support. If a product or product option contains information that is translated to other languages, language support is automatically added for that product or product option. The language supported is either the language for the installed product or, if the product is not installed on this system, the primary language that is installed for OS/400 on this system.

0 Remove the current support information from the system.

Error code

I/O; CHAR(*)

The structure in which to return error information. For the format of the structure, see Error Code Parameter.

Format of Product Information Parameter

The following table shows the format of the product information parameter. For a description of the fields in this format, see Field Descriptions.

Offset		Type	Field
Dec	Hex		
0	0	CHAR(7)	Product ID
7	7	CHAR(6)	Release
13	D	CHAR(4)	Product option
17	11	CHAR(10)	Load ID
27	1B	CHAR(10)	Library

37	25	CHAR(2)	Registration type
39	27	CHAR(14)	Registration value

Field Descriptions

Library.

The name of the principal or main library of the product to be supported. This value is optional. If support is being removed or no library is specified when adding support, this field should contain blanks.

Load ID. The load ID of the product for which support is being added or removed. Load IDs are 4 characters in length; for example, 2924 is the load ID for an English national language version (NLV).

You can use this special value for the load ID:

**CODE* The load ID of the code load for the given product ID, release, and option.

Product ID. The 7-character ID for the product for which support is being added or removed.

Product option. The option number of the product for which support is being added or removed. Use 0000 for the base option. Valid values are 0000 through 0099, where each character is a digit.

Registration type. The registration type associated with the product. This value is optional. If the registration type is not specified, this field should contain blanks. Together, the registration type and registration value make up the registration identifier for the product.

The possible values are:

02 The registration containing a country or region code, city code, and telephone number. All values are 14 characters. This value is the same as specifying *PHONE as the registration type when a product load or product definition is created.

04 The registration value is the same as the registration value for OS/400. The registration value field will be ignored.

08 The registration containing a country or region code and IBM customer number. This value is the same as specifying *CUSTOMER as the registration type when a product load or product definition is created.

Registration value. The registration value associated with the product. This is a 14-character value. If the registration type is not specified, this field should contain blanks. Together, the registration type and registration value make up the registration ID for the product.

Release. The version, release, and modification level of the product for which support is being added or removed. The release must be in the format VxRyMz. Valid values for *x* and *y* are 0 through 9. Valid values for *z* are 0 through 9 or A through Z. For example, V2R1M0 is Version 2, Release 1, Modification 0.

Error Messages

Message ID	Error Message Text
CPF0C1D E	Load ID &1 not valid.
CPF0C2A E	Error occurred while processing QSZSPTPR API.
CPF0C2B E	Code load ID &1 not valid.
CPF0C23 E	Support could be lost for some products.
CPF0C26 E	Length of product information not valid.
CPF0C27 E	Product identifier &1 not valid.
CPF0C28 E	Registration identifier not valid.
CPF0C29 E	Requested action not valid.
CPF0C4A E	Product record not found.
CPF0C4C E	Cannot allocate object &1 in library &2.
CPF0C4E E	Requested product already supported.
CPF0C4F E	Product &1 release &4 option &3 code load must be supported before a language can be supported.
CPF0C50 E	Requested product not supported.
CPF0C8A E	Product option &1 not valid.
CPF0C87 E	Library &1 not allowed.
CPF24B4 E	Severe error while addressing parameter list.
CPF3CF1 E	Error code parameter not valid.
CPF3C21 E	Format name &1 is not valid.
CPF3C29 E	Object name &1 is not valid.
CPF3C90 E	Literal value cannot be changed.
CPF358A E	Release not valid.
CPF9872 E	Program or service program &1 in library &2 ended. Reason code &3.

API introduced: V4R4

Add Product License Information (QLZADDLI) API

Required Parameter Group:			
1	Product identification	Input	Char(*)
2	Product identification format name	Input	Char(8)
3	License information	Input	Char(*)
4	License information format name	Input	Char(8)
5	Error code	I/O	Char(*)
Optional Parameter Group:			
6	Product license information handle	Output	Char(16)
Default Public Authority: *USE			
Threadsafe: No			

The Add Product License Information (QLZADDLI) API adds license information to a product or a feature of a product. License information can be added at two times:

- After the product definition object (*PRDDFN) has been created using the Create Product Definition (CRTPRDDFN) command or the Create Product Definition (QSZCRTPD) API.
- Either before or after the product options have been packaged using the Package Product Option (PKGPRDOPT) command or the Package Product Option (QSZPKGPO) API.

License information must be added before the product is installed.

Authorities and Locks

API QLZADDLI Authority

```
QPGMR(*USE)
QSRV(*USE)
QSRVBAS(*USE)
QSYSOPR(*USE)
```

Product Availability Lock

*SHRRD. The product availability object is in the QUSRSYS library.

Public Authority

*EXCLUDE

Required Parameter Group

Product identification

INPUT; CHAR(*)

Information that uniquely identifies the product or feature to which license information will be added. The structure of this information is determined by the name of the format. For more information, see LICP0100 Format and LICP0200 Format.

Product identification format name

INPUT; CHAR(8)

The name of the format containing the information to identify the product.

The format names are:

LICP0100 Basic product information used as input to the API. For details, see the LICP0100 Format.

LICP0200 Basic product information plus the product feature message ID are used as input to the API. For details, see the LICP0200 Format.

License information

INPUT; CHAR(*)

Information that is used to license the product or feature. The structure of this information is determined by the name of the format. For more information, see LICI0100 Format and LICI0200 Format.

License information format name

INPUT; CHAR(8)

The name of the format containing the license information.

The format names are:

LICI0100 Basic license information used as input to the API. For details, see LICI0100 Format.

LICI0200 Basic license information plus count usage across logical partitions used as input to the API. For details, see LICI0200 Format.

Error code

I/O; CHAR(*)

The structure in which to return error information. For the format of the structure, see Error Code Parameter.

Optional Parameter Group

Product license information handle

OUTPUT; CHAR(16)

A handle based on the product license information. This handle can be used by the software vendor to help ensure asset protection. Specifically, it makes sure that no unauthorized changes were made to the license information. This handle can be used in conjunction with the Retrieve License Information (QLZARTV) API. The information retrieved by the QLZARTV API includes the product license information handle of the product and license information currently installed on the system. If this handle does not match the one returned by this command or API, then the product has been tampered with. It is suggested that this handle be retrieved and checked before doing a request or release when using keyed compliance. This handle is only returned if the product has keyed compliance.

LICP0100 Format

The following information uniquely describes the product or feature for which the license information is to be added. For detailed descriptions of the fields, see Field Descriptions.

Offset		Type	Field
Dec	Hex		
0	0	CHAR(7)	Product ID
7	7	CHAR(6)	Release level
13	D	CHAR(4)	Feature

LICP0200 Format

The following information uniquely describes the product or feature for which the license information is to be added. It also includes the message identifier of the specified feature. For detailed descriptions of the fields, see Field Descriptions.

Offset		Type	Field
Dec	Hex		
0	0	BIN(4)	Size of structure
4	4	CHAR(7)	Product ID
11	B	CHAR(6)	Release level
17	11	CHAR(4)	Feature
21	15	CHAR(7)	Feature message ID

LICI0100 Format

The following specifies the format for the license information that is being added to the product or feature. The allow default usage grace period, vendor password, and grace period fields must be specified if the compliance type is 03. For a compliance type of 01 or 02, these fields do not have to be specified. For detailed descriptions of the fields, see Field Descriptions.

Offset		Type	Field
Dec	Hex		
0	0	CHAR(2)	Usage type
2	2	CHAR(2)	Compliance
4	4	BINARY(4)	Default usage limit
8	8	CHAR(1)	License term
9	9	CHAR(1)	Allow a license to be released
10	A	CHAR(10)	Vendor password
20	14	BINARY(4)	Grace period
24	18	CHAR(1)	Allow default usage limit grace period

LICI0200 Format

The following specifies the format for the license information that is being added to the product or feature. The allow default usage grace period, vendor password, grace period, and count across logical partitions fields must be specified if the compliance type is 03. For a compliance type of 01 or 02, these fields do not have to be specified. For detailed descriptions of the fields, see Field Descriptions.

Offset		Type	Field
Dec	Hex		
0	0	BINARY(4)	Size of structure
4	4	CHAR(2)	Usage type
6	6	CHAR(2)	Compliance
8	8	BINARY(4)	Default usage limit
12	C	CHAR(1)	License term
13	D	CHAR(1)	Allow a license to be released
14	E	CHAR(10)	Vendor password
24	18	BINARY(4)	Grace period
28	1C	CHAR(1)	Allow default usage limit grace period
29	1D	CHAR(1)	Count usage across logical partitions

Field Descriptions

Allow default usage limit grace period. Whether a grace period should be allowed on the default usage limit. If allowed, the default usage limit could be exceeded for the number of days in the grace period for the product or feature. The default usage limit is the number of users that are able to use the product or feature before the license key is installed. Thus, an unlimited number of users can use the product or feature for the number of days in the grace period without having a key.

0 There is no grace period for the default usage limit.

1 There is a grace period for the default usage limit.

Allow a license to be released. Whether a use of the license that was previously requested can be released using the Work with License Information (WRKLICINF) command. When a use of the license is released, the usage count is decremented. This value can only be specified for a usage type of registered.

The valid values are:

0 A use of the license cannot be released with the WRKLICINF command.

1 A use of the license can be released with the WRKLICINF command.

Compliance. The action taken when the usage limit is exceeded.

The valid values are:

01 The usage limit cannot be exceeded. After the usage limit for a product or feature is reached, it cannot be accessed again until the usage limit for it is increased. A message indicating an attempt was made to exceed the usage limit is sent to:

- The QSYSOPR message queue.
- The message queues specified on the Change License Information (CHGLICINF) command.

02 The user who attempts to use the product or feature after the usage limit has been reached is allowed access. A warning message indicating that the usage limit has been exceeded is sent to:

- The QSYSOPR message queue.
- The message queues specified on the CHGLICINF command.

03 To use a product or feature with keyed compliance, the license must be installed using one of the following:

- A valid license key (through the Add License Key Information (ADDLICKEY) command.
- The Add License Key Information (QLZAADDK) API).

This license key is provided by the software provider. The key ties the usage limit to the particular product or feature and to a particular system serial number. To change the usage limit, a user must get a new key from the software provider. A user who attempts to use the product or feature after the usage limit has been reached is allowed access for the number of days contained in the product's grace period. Once the grace period expires, no users over the usage limit are able to use the product or feature until one of the following happens:

- A new license key is received from the software vendor.
- The numbers of users falls below the usage limit.

A warning message, indicating the usage limit is exceeded, is sent to the QSYSOPR message queue and the message queues specified on the CHGLICINF command.

Count usage across logical partitions. Whether usage is counted across logical partitions.

0 Usage is not counted across logical partitions.

1 Usage is counted across logical partitions.

Default usage limit. The usage limit in effect when the product or feature is initially installed.

The valid values are:

0-999999 The number of users allowed to access the product or feature.

-1 Any number of users are allowed to access the product or feature.

Note: For keyed compliance, this is the number of users that will be able to use the product or feature before the license key is installed using the Add License Key command (ADDLICKEY) or API (QLZAADDK). Therefore, to prevent any access to the product or feature without a key, the default usage limit must be set to 0. Also, 0 must be specified for the allow default usage limit grace period.

Feature. The feature of the product to which the license information is being added. Valid values for the feature are 5001 through 9999.

Feature message ID. The message identifier for the message that describes the specified product or feature. If a message ID is not specified, the message identifier used to describe the base product option is used.

Grace period. The number of days after a product first exceeds its usage limit that a user has to obtain a new license key. If a new license key is not obtained from the software provider by the time the grace period is expired, no users over the usage limit are allowed to access the product or feature. When the usage limit is first exceeded, the date the grace period expires is calculated by adding the number of days in the grace period to the current date. Valid values for the grace period are 0 through 999.

During the grace period, the number of users able to use the product or feature is restricted to 50% greater than the current usage limit.

License term. The length of time the authorized usage limit for a product lasts.

The valid values are:

1 The authorized usage limit is valid for the entire version of the product or feature. Each time a new version is installed, the authorized usage limit must be set by using one of the following:

- The Work with License Information (WRKLICINF) command.
- The Change License Information (CHGLICINF) command.

In the case of keyed compliance (type 03), a new license key must also be obtained from the software provider. However, products with keyed compliance (type 03) must set the usage limit by using the ADDLICKEY command. This command also requires that a license key be obtained from the software provider to allow use of the product.

2 The authorized usage limit is valid for the entire release of the product or feature. Each time a new release is installed, the authorized usage limit must be set by using the WRKLICINF command or the CHGLICINF command. In the case of keyed compliance (type 03), a new license key must also be obtained from the software provider. However, products with keyed compliance (type 03) must set the usage limit by using the ADDLICKEY command. This command also requires that a license key be obtained from the software provider to allow use of the product.

3 The authorized usage limit is valid only for a modification of the product. Each time a new modification is installed, the authorized usage limit must be set by using the WRKLICINF command or the CHGLICINF command. However, products with keyed compliance (type 03) must set the usage limit by using the ADDLICKEY command. This command also requires that a license key be obtained from the software provider to allow use of the product.

Product ID. The product ID of the product or feature to which the license information is being added.

Release level. The version, release, and modification level of the product or feature to which the license information is to be added. The release level must be in the format V*x*R*y*M*z*. Valid values for *x* and *y* are 0 through 9. Valid values for *z* are 0 through 9 and A through Z.

Size of structure. The size of the entire data passed in on this structure.

Usage type. The type of license usage.

The valid values are:

01 The usage type is concurrent. It is for the number of unique jobs accessing the product or feature at one time.

02 The usage type is registered. It is for the number of unique users registered by the product or feature.

03 The license usage is by processors. Counts the number of processors that are assigned to the logical partition.

Vendor password. The software vendor's password. This password is encrypted, stored with the product, and is used in validating Add License Key (ADDLICKEY) requests. It must be the same password that is used to generate keys for this product and feature on the Generate License Key command or API. The password must begin with an alphabetic character (A through Z, $, #, or @) followed by no more than 9 alphameric characters (A through Z, 0 through 9, $, #, @, or _).

Error Messages

Message ID	Error Message Text
CPF0CB2 E	Product identifier &1 not valid.
CPF0C4B E	Product availability object &2/&1 recovery required.
CPF0C4C E	Cannot allocate object &1 in library &2.
CPF0C4D E	Error occurred while processing object &1 in library &2.
CPF0C54 E	Data in product record not correct.
CPF24B4 E	Severe error while addressing parameter list.
CPF3CF1 E	Error code parameter not valid.
CPF3C21 E	Format name &1 is not valid.
CPF3C90 E	Literal value cannot be changed.

CPF358A E	Release not valid.
CPF8191 E	Product definition &4 in &9 damaged.
CPF8193 E	Product load object &4 in &9 damaged.
CPF9E0A E	Allow license release not valid.
CPF9E0B E	Allow grace period value of &1 not valid.
CPF9E0C E	Allow license release &1 not valid.
CPF9E0D E	Grace period &1 not valid.
CPF9E0E E	Feature message identifier &1 not valid.
CPF9E0F E	Vendor password &1 not valid.
CPF9E02 E	Cannot add license information.
CPF9E03 E	License information already exists.
CPF9E04 E	Product definition for product &1 &2 feature &3 cannot be found.
CPF9E05 E	Feature &3 not valid.
CPF9E06 E	Usage type &1 not valid.
CPF9E07 E	Compliance &1 not valid.
CPF9E08 E	Default usage limit &1 not valid.
CPF9E09 E	License term &1 not valid.
CPF9E1A E	License information conflict found.
CPF9E15 E	Error in license management function.
CPF9801 E	Object &2 in library &3 not found.
CPF9803 E	Cannot allocate object &2 in library &3.
CPF9810 E	Library &1 not found.
CPF9838 E	User profile storage limit exceeded.
CPF9872 E	Program or service program &1 in library &2 ended. Reason code &3.
CPF9872 E	Program or service program &1 in library &2 ended. Reason code &3.

API Introduced: V2R3

Check Target Release (QSZCHKTG) API

Required Parameter Group:

1	Target release	Input	Char(10)
2	Supported OS/400 releases list	Input	Char(*)
3	Number of supported OS/400 releases	Input	Binary(4)
4	Validated target release	Output	Char(6)
5	Equivalent supported OS/400 release	Output	Char(6)
6	Error code	I/O	Char(*)

Default Public Authority: *USE

Threadsafe: No

The Check Target Release (QSZCHKTG) API verifies that you specified a valid target release value. You can use this API to check a target release value for CL commands that support a TGTRLS keyword parameter. If the target release is a known OS/400 release, this API returns the validated target release. If you specify *CURRENT or *PRV for the target release value, this API returns the equivalent VxRxMx release value.

This API also determines, from the supported OS/400 releases list parameter, which supported release is the most recent release that is less than or equal to the validated target release.

Authorities and Locks

None

Required Parameter Group

Target release

INPUT; CHAR(10)

The target release value being checked. The value must be *CURRENT, *PRV, or a 6-character release value in the format VxRxMx where *x* is a digit from 0 through 9. The 6-character release value must be a known release of OS/400. All values must be left-aligned and padded with blanks.

Supported OS/400 releases list

INPUT; CHAR(*)

The list of OS/400 releases that the caller of this API supports. Each list entry must be a 6-character release value in the format VxRxMx (where *x* is a digit from 0 through 9) or the special value *SAV.

A 6-character release value must be a known release of OS/400. All values must be unique and must be in ascending order. If V3R2M0 and V3R6M0 are both in the list, V3R2M0 must precede V3R6M0. The earliest valid OS/400 release value is V1R3M0.

Assume that the calling function supports the creation of two different versions of an object. The first version can be used on a system that runs OS/400 V3R1M0 (or any later release). The second version can only be used on a system that runs OS/400 V3R7M0 (or any later release). The function would pass V3R1M0V3R7M0 for this parameter and would pass a value of 2 for the number of supported OS/400 releases parameter.

If *SAV is specified, it must be the only entry in the list. This special value indicates that the caller supports the same set of OS/400 releases as the OS/400 Save Library (SAVLIB) and Save Object (SAVOBJ) commands.

Number of supported OS/400 releases

INPUT; BINARY(4)

The number of list entries passed in the supported OS/400 releases list parameter. This parameter must have a value of 1 or greater. The value must be 1 if *SAV is specified in the supported OS/400 releases list.

Validated target release

OUTPUT; CHAR(6)

The validated target release being returned. If the input target release is *CURRENT or *PRV, this parameter contains the equivalent VxRxMx value. If the input target release is a known OS/400 release, the API copies the value to this parameter and returns the value. If the input target release value is not *CURRENT, *PRV, or a known OS/400 release in VxRxMx format, the API returns blanks for this parameter.

Equivalent supported OS/400 release

OUTPUT; CHAR(6)

The release list entry being returned, from the supported OS/400 releases list parameter, that is the most recent release that is less than or equal to the validated target release. If the input target release value is not valid, the API returns blanks. If the input target release is less than the first list entry in the supported OS/400 releases list parameter, the API sends message CPF0C35 and returns blanks.

If *SAV is specified in the supported OS/400 releases list, the API returns the validated target release.

Error code

I/O; CHAR(*)

The structure in which to return error information. For the format of the structure, see Error Code Parameter.

Error Messages

Message ID	Error Message Text
CPF0C31 E	&1 not valid for number of supported OS/400 releases.
CPF0C32 E	List of supported OS/400 releases not valid.
CPF0C33 E	Target release &1 not valid.
CPF0C34 E	Release &1 is not a valid OS/400 release.

CPF0C35 E	Target release &1 is not a supported release.
CPF24B4 E	Severe error while addressing parameter list.
CPF3CF1 E	Error code parameter not valid.

API Introduced: V3R7

Top | Software Product APIs | APIs by category

Copy Program Temporary Fix to Save File (QPZCPYSV) API

Required Parameter Group:

1	Product information	Input	Char(*)
2	Product information format name	Input	Char(8)
3	Device name	Input	Char(10)
4	Error code	I/O	Char(*)

Default Public Authority: *EXCLUDE

Threadsafe: No

The Copy Program Temporary Fix to Save File (QPZCPYSV) API allows you to copy all program temporary fixes (PTFs) for the selected product from the media and store them in *SERVICE. The PTFs can then be displayed, loaded, or copied using the PTF commands. » PTF Group information can also be copied in addition to the PTFs. «

A save file is created in library QGPL for each PTF that is copied. You can use the Retrieve Program Temporary Fix Information (QPZRTVFX) API to obtain the name of the save file.

If the PTF already exists in *SERVICE, the PTF is not copied.

Cover letters are copied to file QAPZCOVER in library QGPL.

Authorities and Locks

Device

*USE

Library QGPL

*USE

Lock conflicts may occur if this API is called while another PTF » or PTF group « operation is in progress.

Required Parameter Group

Product information

INPUT; CHAR(*)

The information needed to put the PTF into the *SERVICE device.

Product information format name

INPUT; CHAR(8)

The name of the format that describes the product information. The only format name supported is:

PTFV0100 See PTFV0100 Format.

Device name

INPUT; CHAR(10)

The name of the optical device or tape device that contains the PTFs to be copied.

Error code

I/O; CHAR(*)

The structure in which to return error information. For the format of the structure, see Error Code Parameter.

PTFV0100 Format

The following table describes the format for the product information parameter. The format identifies the product of the PTFs to be copied. For detailed descriptions of the fields, see Field Descriptions.

Offset		Type	Field
Dec	Hex		
0	0	BINARY(4)	Length of product information
4	4	CHAR(7)	Product ID
11	B	CHAR(6)	Release
17	11	CHAR(1)	Automatically support product
»18	12	CHAR(1)	Copy PTFs
19	13	CHAR(1)	Copy PTF Groups «

Field Descriptions

Automatically support product. Indicates that when a PTF for a product currently not supported or installed on the system is found on the media, the product automatically will be marked as supported and the PTF will be copied. If the PTF affects a National Language Version, only the primary language of the system will be supported automatically. You can use the Add or Remove Product Support (QSZSPTPR) API to add support for additional languages.

This field is ignored when *SUPPTD is specified in the product ID field.

0 Do not automatically support the product

1 Automatically support the product

»**Copy PTFs.** Indicates which PTFs to copy. If this field is not specified, a value of 0 is assumed.

0 Copy all PTFs regardless of their status.

1 Copy save files only for PTFs that do not have a status of superseded, temporarily applied, or permanently applied on your system.

«

»**Copy PTF Groups.** Indicates if the information for PTF Groups are to be copied in addition to the PTFs. The PTF Groups can then be displayed using the WRKPTFGRP command or retrieved using the QpzListPtfGroupDetails API. Note this field only applies when the Product ID field specifies *ALL or *SUPPTD. If this field is not specified, a value of 0 is assumed.

0 Do not copy the PTF Group information.

1 Copy an existing PTF Group of the same name only when the level of the PTF Group being copied is higher than the level of the PTF Group on the system.

2 Copy an existing PTF Group of the same name only when the level of the PTF Group being copied is equal to or higher than the level of the PTF Group on the system.

3 Always copy the PTF Group information. If a PTF Group of the same name already exists on the system, it will be replaced.

«

Length of product information. The length of data in the product information format, including this field. The only valid values for this field are 18 or 20.

Product ID. The name of the licensed products for which PTFs are to be copied. You can use the following special values for the product ID:

**ALL* PTFs for all products, releases, options, and languages that are either supported or installed on this system are copied from the media.

Caution should be used when specifying this value and indicating automatic support for the product. PTF media frequently contains PTFs for all products available from your service provider. If you specify these values together, all PTFs from the media are copied on to your system.

**SUPPTD* PTFs for all products, releases, options, and languages that are currently supported on this system are copied. PTFs for products that are installed but not supported are not copied.

Release. The version, release, and modification of the product. The release must be in the format V*x*R*y*M*z*. Valid values for *x* and *y* are 0 through 9. Valid values for *z* are 0 through 9 or A through Z. You can use the following special value for the release:

**ALL* PTFs for all releases of the product will be copied.

A value other than *ALL is allowed only when a particular product is specified for the product ID field.

Error Messages

CPF24B4 E	Severe error while addressing parameter list.
CPF35BE E	Product &1 &3 not supported or installed.
CPF35CC E	Library required for building PTFs already exists.
CPF3C21 E	Format name &1 is not valid.
CPF3C4B E	Value not valid for field &1.
CPF3C4C E	Value not valid for field &1.
CPF3CF1 E	Error code parameter not valid.
CPF358A E	Release not valid. »
CPF3598 E	PTF operation already in progress.
CPF36AF E	PTF group operation already in progress. «
CPF9814 E	Device &1 not found.
CPF9820 E	Not authorized to use library &1.
CPF9825 E	Not authorized to device &1.
CPF9872 E	Program or service program &1 in library &2 ended. Reason code &3.

API introduced: V4R4

Top | Software Product APIs | APIs by category

Create Product Definition (QSZCRTPD) API

Required Parameter Group:

1	Qualified product definition name	Input	Char(20)
2	Product definition information	Input	Char(106)
3	Product option list	Input	Array of Char(41)
4	Number of product options	Input	Binary(4)
5	Language load list	Input	Array of Char(20)
6	Number of language loads	Input	Binary(4)
7	Text description	Input	Char(50)
8	Public authority	Input	Char(10)
9	Error code	I/O	Char(*)

Default Public Authority: *USE

Threadsafe: No

The Create Product Definition (QSZCRTPD) API creates a product definition (*PRDDFN) object. Each release of a packaged software product requires one product definition.

Authorities and Locks

Library Authority

*ADD and *READ

Library Lock

*SHRUPD

Product Availability Lock

*SHRRD. The product availability object resides in the QUSRSYS library.

Required Parameter Group

Qualified product definition name

INPUT; CHAR(20)

The first 10 characters contain the product definition name. The second 10 characters contain the name of the library into which the product definition is to be created.

The following special value is supported for the library name:

*CURLIB The job's current library

Product definition information

INPUT; CHAR(106)

A structure containing information about the product. For more information, see Format of Product Definition Information.

Product option list

INPUT; ARRAY of CHAR(41)

An array containing information for each of the options defined for the product. Each element of the array contains information for one option. There must be one element of the array for each option. The first element of the array must be for the base (0000) option. The required data for the product option list is described in Format of Product Option List.

Number of product options

INPUT; BINARY(4)

The number of options defined for the product. This number is the same as the number of elements in the product option list parameter. Up to 100 product options can be specified. If the number of elements in the product option list is less than the value specified, the results are unpredictable.

Language load list

INPUT; ARRAY of CHAR(20)

Specifies which languages are defined for the product options. The required data for the language load list is described in Format of Language Load List.

Number of language loads

INPUT; BINARY(4)

The number of elements in the language load list parameter. If the number of elements in the language load list parameter is less than the value specified, the results are unpredictable. The valid range is 1 to 139.

Text description

INPUT; CHAR(50)

Text that briefly describes the product definition object.

Public authority

INPUT; CHAR(10)

The authority you give to users who do not have specific authority to the product definition object and whose group profile has no specific authority to the object.

Valid values are:

*ALL	Allows the user to perform all operations on the object except those limited to the owner or controlled by the authorization list management authority.
*CHANGE	Allows the user to perform all operations on the object except those limited to the owner or controlled by the object existence authority and object management authority.

*EXCLUDE — Prevents the user from accessing the object.

*LIBCRTAUT — The public authority for the object is taken from the value of the create authority (CRTAUT) parameter of the target library. (This is the library that is to contain the object.) This value is determined when the object is created. If the CRTAUT value for the library changes after the object is created, the new value does not affect any existing objects.

*USE — Provides object operational authority and read authority.

Error code

I/O; CHAR(*)

The structure in which to return error information. For the format of the structure, see Error Code Parameter.

Format of Product Definition Information

The following table describes the order and format of the product definition information parameter. For detailed descriptions of fields in the table, see the Field Descriptions.

Offset		Type	Field
Dec	Hex		
0	0	CHAR(7)	Product ID
7	7	CHAR(6)	Release level
13	D	CHAR(10)	Message file
23	17	CHAR(10)	First copyright
33	21	CHAR(10)	Current copyright
43	2B	CHAR(6)	Release date
49	31	CHAR(4)	Allow multiple releases
53	35	CHAR(10)	Registration ID type
63	3F	CHAR(14)	Registration ID value
77	4D	CHAR(1)	Release date century
78	4E	CHAR(28)	Reserved

Format of Product Option List

Up to 100 product options can be specified. The first option specified must be 0000 (the base option). The product option list parameter is described in the table below. The offsets shown in the table are for the first element in this array. For detailed descriptions of fields in the table, see the Field Descriptions.

Offset		Type	Field
Dec	Hex		

0	0	CHAR(4)	Product option number
4	4	CHAR(7)	Message ID
11	B	CHAR(10)	Allow dynamic naming
21	15	CHAR(4)	Code load ID
25	19	CHAR(16)	Reserved

Format of Language Load List

Up to 40 language loads can be specified for the base (0000) option. A maximum of 139 total can be specified. The language load list field is described in the table below. The offsets shown in the table are for the first element in this array. For detailed descriptions of fields in the table, see the Field Descriptions.

Offset		Type	Field
Dec	**Hex**		
0	0	CHAR(8)	Language load ID
8	8	CHAR(4)	Product option number
12	C	CHAR(8)	Reserved

Field Descriptions

Allow dynamic naming. Allow libraries and root folders to be named during installation of the product option.

Valid values are:

**NODYNNAM* Do not allow naming of libraries and folders during installation.

**ALWDYNNAM* Allow naming of libraries and root folders during installation.

Allow multiple releases. Allow different releases of this product to be installed on the same system.

Valid values are:

**NO* Do not allow multiple releases.

**YES* Allow multiple releases.

Code load ID. The identifier of the code load for the option. Valid values are 5001 through 9999. For all product options that are part of the same feature, specify the same code load ID. If you are not adding license information to your product, you should use 5001 for the code load ID for each option. The code load ID you specify for this option must be the same as the load ID specified when you create the code product load for this option. See Create Product Load (QSZCRTPL) API for information about creating product loads.

Current copyright. The year of the most recent copyright for this product. The year must be specified as a 4-digit number, such as 1990. The field must be padded with blanks. If this field and the first copyright field have values other than *NONE, the first copyright field must be less than the current copyright field.

The following special values are valid:

**CURRENT* The current year is used.

**NONE* The product does not have a current copyright.

First copyright. The first year that the product was copyrighted. The year must be specified as a 4-digit number, such as 1990. The field must be padded with blanks.

The following special values are valid:

**CURRENT* The current year is used.

**NONE* The product does not have a first copyright.

Language load ID. The national language versions (NLVs) that are valid for a given product option. Individual NLVs may be specified for the base option, with one element of the language load list parameter for each NLV for the base option. For options other than the base option, only *BASEOPT and *NONE are valid. For example, to create a product definition with NLVs 2924 and 2931 defined for both the base option and option 1, the language load list would have three elements:

- One with 2924 0000.
- One with 2931 0000.
- One with *BASEOPT 0001.

Valid special values are:

**NONE* Defines the option to have no NLVs.

**IBMLNG* Only valid for the base option, 0000. Specifies the product definition is created with the list of all the NLVs for the base option the same as the currently installed operating system.

**BASEOPT* Only valid for an option other than 0000. Defines this option to have the same NLVs as the base option of this product.

Message file. The name of the message file containing the messages that describe the product and its options. The message file for the base option is considered the message file for the product.

Message ID. The identifier of the message that describes the product option.

Product ID. The 7-character identifier of the product for which a product definition is being created. The product ID must be in the format *nlxxxxx*, where *n* is any numeric character 0 through 9. The *l* is any uppercase letter A through Z, and *x* is any numeric character 0 through 9 or uppercase letter A through Z.

Product option number. The identifier of the product option.

When used in the product option list parameter, valid values are 0000 through 0099, with each number specified at most once. Specify 0000 for the base product option. The value 0000 must be the first option specified.

When used in the language load list parameter, this is the identifier for the product option for which NLVs are being defined. This must be one of the options specified on the product option list parameter.

Registration ID type. Specifies what the registration ID value field represents.

Valid values are:

*PHONE Telephone number will be entered in the registration ID value field.

*CUSTOMER Country or region code and IBM customer number will be entered in the registration ID value field.

Registration ID value. The identifier of the organization to which the product belongs. This number should be unique from other vendors on the systems on which this product will be installed. It is recommended you specify a telephone number (including the country or region code and city code) or your IBM customer number appended to your country or region code. Valid characters for the registration ID value are A through Z and 0 through 9, padded with blanks on the right.

Release date. Release date of the product. The format is yymmdd, where yy is the year, mm is the month, and dd is the day.

The following special value is valid for the release date:

*NONE The product does not have a release date.

Release date century. The century that corresponds to the release date of the product. This field is ignored if *NONE is specified for release date.

Possible values follow:

0 Indicates years 19*xx*

1 Indicates years 20*xx*

Blank The release date century is set to 0 if the release date year is equal to or greater than 40 (years 1940 through 1999); it is set to 1 if the release date year is less than 40 (years 2000 through 2039).

Release level. The version, release, and modification level of the product being created in the format V*x*R*y*M*z*. Valid values for *x* and *y* are 0 through 9. Valid values for *z* are 0 through 9 or A through Z. For example, V2R1M0 is Version 2, Release 1, Modification 0.

Reserved. This field must contain blank characters; otherwise, an error occurs.

Error Messages

Message ID	Error Message Text
CPF0CAA E	First product option not *BASE.
CPF0CAB E	Language load identifier not valid.
CPF0CAC E	Language load identifier not valid.
CPF0CAD E	Duplicate language load identifier specified.
CPF0CAE E	Language load not valid.
CPF0CAF E	Duplicate option &3 specified.
CPF0CA3 E	Code load &4 supported.

CPF0CA6 E Product definition &3 not created in library &4.

CPF0CBA E Number of options parameter not valid.

CPF0CBB E Release date &3 not valid.

CPF0CBC E Message identifier &4 not allowed.

CPF0CBD E Code load ID &9 not valid.

CPF0CBE E Release date century field not valid.

CPF0CB0 E Copyright dates not valid.

CPF0CB1 E Registration identifier not valid.

CPF0CB2 E Product identifier &1 not valid.

CPF0CB3 E Value for reserved field not valid.

CPF0CB5 E Copyright field &3 not valid.

CPF0CB6 E Allow multiple releases field not valid.

CPF0CB7 E Allow dynamic naming field not valid.

CPF0CB8 E Language load field not valid.

CPF0CB9 E Number of language loads parameter not valid.

CPF0C16 E Object &1 type &3 already exists in library &2.

CPF0C17 E *&3 object already exists for product &4 release &5.

CPF0C18 E Registration identifier &7 not valid for product &4 release &5.

CPF0C19 E Damage occurred on object &1 in library &2.

CPF0C4A E Product record not found.

CPF0C4D E Error occurred while processing object &1 in library &2.

CPF0C8A E Product option &1 not valid.

CPF0C84 E Load identifier &4 not valid.

CPF0C86 E Registration identifier not valid.

CPF0C9B E Authority &1 not valid.

CPF24B4 E Severe error while addressing parameter list.

CPF3CF1 E Error code parameter not valid.

CPF3C29 E Object name &1 is not valid.

CPF3C90 E Literal value cannot be changed.

CPF35E3 E Interface error detected.

CPF358A E	Release not valid.
CPF9810 E	Library &1 not found.
CPF9818 E	Object &2 in library &3 not created.
CPF9819 E	Object &2 in library &3 not created.
CPF9820 E	Not authorized to use library &1.
CPF9830 E	Cannot assign library &1.
CPF9872 E	Program or service program &1 in library &2 ended. Reason code &3.

API Introduced: V4R4

Top | Software Product APIs | APIs by category

Create Product Load (QSZCRTPL) API

Required Parameter Group:

1	Product load name	Input	Char(10)
2	Product load information	Input	Char(87)
3	Secondary language library name	Input	Char(10)
4	Principal library information	Input	Char(30)
5	Additional library list	Input	Array of Char(30)
6	Number of additional libraries	Input	Binary(4)
7	Preoperation exit programs	Input	Array of Char(20)
8	Number of preoperation exit programs	Input	Binary(4)
9	Folder list	Input	Array of Char(126)
10	Number of folders	Input	Binary(4)
11	Text description	Input	Char(50)
12	Public authority	Input	Char(10)
13	Error code	I/O	Char(*)

Optional Parameter Group 1:

14	Directory list	Input	Array of Char(*)
15	Number of directories	Input	Binary(4)
16	Directory list format name	Input	Char(8)

»Optional Parameter Group 2:

17	Software agreement document list	Input	Array of Char(*)
18	Number of software agreement documents	Input	Binary(4)«

Default Public Authority: *EXCLUDE

Threadsafe: No

The Create Product Load (QSZCRTPL) API creates a product load (*PRDLOD) object. Each release of a software product requires one or more product load objects.

Authorities and Locks

Library Authority

*ADD and *READ

Library Lock

*SHRUPD

Product Availability Lock

*SHRRD. The product availability object resides in the QUSRSYS library.

Required Parameter Group

Product load name

INPUT; CHAR(10)

The name of the product load object to be created. The product load is created into the principal development library. The following special value is valid:

**LNG* The name of the load object is the same as the previously created language load object for this product; version, release, and modification level; and option. This special value is only valid if a language product load is being created.

Product load information

INPUT; CHAR(87)

A structure containing information about the product load. For more information, see the Format of Product Load Information.

Secondary language library name

INPUT; CHAR(10)

The name of the secondary language library for the language product load being created. This is the library into which this product load is installed if:

- The language identifier for this product load does not match the system primary language identifier.
- No override name is specified on the Restore Licensed Program (RSTLICPGM) command.

This field is valid only if a language product load is being created.

Principal library information

INPUT; CHAR(30)

The first 10 characters specify the principal development library. The second 10 characters specify the principal primary library. The last 10 characters specify the postoperation exit program for both principal libraries. For more information, see Format of Principal Library Information.

Additional library list

INPUT; ARRAY of CHAR(30)

The additional libraries for the product load. Additional libraries do not need to exist before the product load object is created.

For each element:

- The first 10 characters specify the development library.
- The second 10 characters specify the primary library.

- The last 10 characters specify the postoperation exit program for both libraries.

For more information, see Format of Additional Library List.

Number of additional libraries

INPUT; BINARY(4)

The number of elements in the additional library list. If the number of elements in the additional library list is less than the value specified, the results are unpredictable.

Preoperation exit programs

INPUT; ARRAY of CHAR(20)

The preoperation exit programs for this load. The first 10 characters specify the exit program name. The second 10 characters specify the development library name. For more information, see Format of Preoperation Exit Programs.

Number of preoperation exit programs

INPUT; BINARY(4)

The number of elements in the preoperation exit programs array. If the number of elements in the preoperation exit programs parameter is less than the value specified, the results are unpredictable.

Folder list

INPUT; ARRAY of CHAR(126)

The folders for this product load. When creating a code load, the first folder specified must be a root folder. When creating a language load, the first folder specified must be a subfolder of a root folder. The folders do not need to exist before the product load object is created.

Each product option has at most one root folder. A folder cannot belong to more than one product option. The root folder must be part of the code load. Folders must be specified so that a parent folder precedes its subfolder on the list.

The number of folders must be zero if the number of directories names in the directory list parameter is greater than zero.

A maximum of 100 folders can be specified for a load.

The documents in the development folders are saved when the product load is saved with the Save Licensed Program (SAVLICPGM) command. For more information, refer to the System Manager Use book .

For each element of the folder list, the first 63 characters specify the development folder and the next 63 characters specify the primary folder. For more information, see Format of Folder List.

Number of folders

INPUT; BINARY(4)

The number of elements in the folder list array. If the number of elements in the folder list is less than the value specified, the results are unpredictable.

Text description

INPUT; CHAR(50)

Text that briefly describes the product load object.

Public authority

INPUT; CHAR(10)

The authority you give to users:

- Who do not have specific authority to the product load object.
- Whose group profile has no specific authority to the object.

Valid values are:

**ALL*	Allows the user to perform all operations on the object except those limited to the owner or controlled by the authorization list management authority.
**CHANGE*	Allows the user to perform all operations on the object except those: ❍ Limited to the owner. ❍ Controlled by the object existence authority and object management authority.
**EXCLUDE*	Prevents the user from accessing the object.
**LIBCRTAUT*	The public authority for the object is taken from the value of the create authority (CRTAUT) parameter of the target library. (This is the library that is to contain the object). This value is determined when the object is created. If the CRTAUT value for the library changes after the object is created, the new value does not affect any existing objects.
**USE*	Provides object operational authority and read authority.

Error code

I/O; CHAR(*)

The structure in which to return error information. For the format of the structure, see Error Code Parameter.

Optional Parameter Group 1

Directory list

INPUT;CHAR(*)

The directories for this product load. You cannot specify directory names if any folder names are specified in the folder list parameter. A maximum of 300 directory full path names can be specified for a load. For more information, see DIRI0100 Format.

Number of directories

INPUT;BINARY(4)

The number of elements in the directory list array. If the number of elements in the directory list is less than the value specified, the results are unpredictable.

Directory list format name

INPUT; CHAR(8)

The name of the format containing the directory list. The format name is:

DIRI0100 See DIRI0100 Format.

Optional Parameter Group 2

Software agreement document list

INPUT; CHAR(*)

The software agreement documents for this product option. The software agreement documents do not need to exist before the product load object is created. The software agreement documents must be created into a directory in '/QIBM/UserData/LicenseDoc' prior to successfully packaging the product option. For more information, see Format of Software Agreement Document List.

Number of software agreement documents

INPUT; BINARY(4)

The number of elements in the software agreement document list array. There must be a minimum of one and a maximum of ten for software agreements to be enabled for a product load. If the number of elements in the software agreement document list is less than the value specified, the results are unpredictable.

Format of Product Load Information

The product load information parameter is described in the following table. For a detailed description of the fields in the table, see Field Descriptions.

Offset		Type	Field
Dec	Hex		
0	0	CHAR(7)	Product ID
7	7	CHAR(6)	Release level
13	D	CHAR(4)	Product option
17	11	CHAR(10)	Product load type
27	1B	CHAR(8)	Load ID
35	23	CHAR(10)	Registration ID type
45	2D	CHAR(14)	Registration ID value
59	3B	CHAR(10)	Minimum target release
69	45	CHAR(18)	Reserved

Format of Principal Library Information

The principal library information parameter is described in the following table. For a detailed description of the fields in the table, see Field Descriptions.

Offset		Type	Field
Dec	Hex		
0	0	CHAR(10)	Principal development library name
10	A	CHAR(10)	Principal primary library name
20	14	CHAR(10)	Postoperation exit program name

Format of Additional Library List

The following table describes the additional library list parameter. The offsets shown in the table are for the first element in this array. For a detailed description of the fields in the table, see Field Descriptions.

Offset		Type	Field
Dec	Hex		
0	0	CHAR(10)	Additional development library name
10	A	CHAR(10)	Additional primary library name
20	14	CHAR(10)	Postoperation exit program name

Format of Preoperation Exit Programs

The following table describes the preoperation exit programs parameter. The offsets shown in the table are for the first element in this array. For a detailed description of the fields in the table, see Field Descriptions.

Offset		Type	Field
Dec	Hex		
0	0	CHAR(10)	Preoperation exit program name
10	A	CHAR(10)	Development library name

Format of Folder List

The following table describes the folder list parameter. The offsets shown in the table are for the first element in this array. For a detailed description of the fields in the table, see Field Descriptions.

Offset		Type	Field
Dec	Hex		
0	0	CHAR(63)	Development folder

63	3F	CHAR(63)	Primary folder

DIRI0100 Format

The following table describes the directory list parameter. The offsets shown in the table are for the first element in this array. The decimal and hexadecimal offsets to subsequent entries are determined by using the length of the full path length field, the length of the home directory length field, and the value of the full path length field. For a detailed description of the fields in the table, see Field Descriptions.

Offset		Type	Field
Dec	Hex		
0	0	BINARY(4)	Full path length
4	4	BINARY(4)	Home directory length
8	8	CHAR(*)	Full path name

Format of Software Agreement Document List

The following table describes the software agreement document list parameter. The offsets shown in the table are for the first element in this array. For a detailed description of the fields in the table, see Field Descriptions.

Offset		Type	Field
Dec	Hex		
0	0	BINARY(4)	Software agreement document length
8	8	CHAR(*)	Software agreement document name

Field Descriptions

Additional development library name. The name of the additional development library.

Additional primary library name. The additional primary library name. Valid special values are:

**DVLLIB* The development library name is used as the primary library name.

**CODE* The additional primary library in the code load corresponding to the immediately preceding development library is used. This value is valid only when the product load type is specified as *LNG in the product load information parameter.

Development folder. The name of a development folder for this load. The folder does not need to exist before the product load object is created.

Development library name. The development library with which the preoperation exit program is associated. The preoperation exit program may be associated with the principal library or an additional library. If associated with the principal library, this must be the same as the value for the principal development library in the

principal library information parameter. If associated with an additional library, this must be the same as one of the values for the additional development library in the additional library list parameter. Valid special values are:

**PRDDFN* Specify this value if:

- You want the preoperation exit program to be associated with the principal development library.
- You specified *PRDDFN for the principal development library field of the principal library information parameter.

  This is only valid when *PRDDFN is specified for the principal development library in the principal library information parameter.

**CODE* Specify this value if you want the preoperation exit program to be associated with the principal development library and if you specified *CODE as the principal development library field of the principal library information parameter. This is only valid when *CODE is specified for the principal development library in the principal library information parameter.

Full path length. The length of the full path name, in bytes.

Full path name. The fully qualified directory name assigned to this product load. Naming restrictions for directories assigned to a product load include:

- You cannot specify /QSYS.LIB and /QDLS directories.
- You must specify unique full path names.
- You cannot end the path name with a forward slash.
- You cannot use the "." or ".." directories in the path name.
- You cannot use "//" in the directory path name.

Home directory length. The length of the home directory part of the full path name.

Load ID. The load ID of the product load to be created. For language loads, this must be a valid national language version (NLV). The following special value is valid:

**CODEDFT* The default code load ID, 5001, is used. This value is valid only when the product load type field is *CODE.

Minimum target release. The minimum release of the operating system to which the SAVLICPGM command allows the product to be saved. The format is V*x*R*y*M*z*. Valid values for *x*, *y*, and *z* are 0 through 9. For example, V3R1M0 is Version 3, Release 1, Modification 0. If this field is blank, the version, release, and modification level of the operating system is used. This value may be different for each load, however, the code load must specify the earliest release for a given option. Also, the code load for the base option must specify the earliest release for a given product. Valid special values are:

**CURRENT* The version, release, and modification level of the operating system is used.

**PRV* The version, release, and modification level previous to that of the operating system is used. Previous is the previous release with modification level 0 of the operating system.

**CODE* The minimum target release of the code load for this option is used.

**BASECODE* The minimum target release of the code load for the base option is used.

Postoperation exit program name. The program that is called in the corresponding installed library after any

of these operations are performed on the product load:

- Save, using the Save Licensed Program (SAVLICPGM) command
- Restore, using the Restore Licensed Program (RSTLICPGM) command
- Check, using the Check Product Option (CHKPRDOPT) command

The exit program does not need to exist before the product load object is created. The following special value is valid:

**NONE* No exit program is called after the library is saved, restored, or checked.

Preoperation exit program name. The name of the preoperation exit program. A preoperation exit program is called before any of these operations are performed on the library:

- Save, using the Save Licensed Program (SAVLICPGM) command
- Restore, using the Restore Licensed Program (RSTLICPGM) command
- Delete, using the Delete Licensed Program (DLTLICPGM) command.

The exit program does not need to exist before the product load object is created.

Primary folder. The primary folder name associated with the development folder. The following special value is valid:

**DVLFLR* The development folder name is the same as the primary folder name.

Principal development library name. The library into which the product load is created. Valid special values are:

**PRDDFN* The name of the library in which the product definition exists is used for the development library name.

**CODE* The name of the principal development library for the code load is used. This value is valid only when the product load type field is specified as *LNG.

Principal primary library name. The product load is installed into this library when no override name is specified on the Restore Licensed Program (RSTLICPGM) command. Valid special values are:

**DVLLIB* The development library name is used as the primary library name.

**CODE* The name of the principal development library for the code load is used. This value is valid only when the product load type field is specified as *LNG.

Product ID. The 7-character identifier of the product for which a product load is being created. The product ID must be in the format *nlxxxxx*, where *n* is any numeric character 0 through 9. The *l* is any uppercase letter A through Z, and *x* is any numeric character 0 through 9 or uppercase letter A through Z.

Product load type. Whether the product load being created is a code load or a language load. Valid values are:

**CODE* A code load is created.

**LNG* A language load is created.

Product option. The product option for which a product load is being created. Use 0000 for the base option.

Registration ID type. Specifies what the registration ID value field represents. Valid values are:

**PRDDFN*	The registration ID is taken from the product definition for this product and release level. The product definition must exist for *PRDDFN to be valid.
**PHONE*	A telephone number will be entered in the registration ID value field.
**CUSTOMER*	The country or region code and IBM customer number will be entered in the registration ID value field.

Registration ID value. Identifier of the organization to which the product belongs. This number should be unique from other vendors on the systems on which this product will be installed. It is recommended that you specify a telephone number, including the country or region and city code, or specify your country or region code followed by your IBM customer number. Valid characters for the registration ID value are A through Z and 0 through 9, padded with blanks on the right.

Release level. The version, release, and modification level of the product being created in the format V*x*R*y*M*z*. Valid values for *x* and *y* are 0 through 9. Valid values for *z* are 0 through 9 or A through Z. For example, V2R1M0 is Version 2, Release 1, Modification 0.

Reserved. This field must contain blank characters; otherwise, an error occurs.

Software agreement document length. The length of the software agreement document in bytes.

Software agreement document name. The name of the software agreement document. For products using software agreements, the software agreement documents do not have to exist at the time the product load is created but must exist when the product option is packaged and must reside in a specific directory structure in IFS. The software agreement document repository is '/QIBM/UserData/LicenseDoc'. Each software agreement document must have its own subdirectory under '/QIBM/UserData/LicenseDoc'. The subdirectory must be named the same as the software agreement document. Each of these subdirectories must contain the actual software agreement document(s), translated for your supported languages, stored in UTF 16 (Big Endian). See Generate online software agreements for instructions on creating software agreements in UTF 16. Additionally, each document name within the subdirectory must contain the appropriate language extension that matches the supported languages. See Approved Language Suffixes for a full list.

The following example provides guidance concerning the naming of software agreement documents:

Assumptions:

- The product load being created is going to be enabled for software agreements.
- The product load being created includes three software agreement documents, translated in English and French, that need to be accepted prior to the product option being successfully installed on the system.
- The software agreement documents for this product load are named as follows:
    1. document 1: 1MYPROD-V7R4M1-0000-01_en (English translation)
       1MYPROD-V7R4M1-0000-01_fr (French translation)
    2. document 2: 1MYPROD-V7R4M1-0000-02_en (English translation)
       1MYPROD-V7R4M1-0000-02_fr (French translation)
    3. document 3: 1MYPROD-V7R4M1-0000-03_en (English translation)
       1MYPROD-V7R4M1-0000-03_fr (French translation)

Actions to perform:

1. In the '/QIBM/UserData/LicenseDoc' directory create three sub-directories with the names '1MYPROD-V7R4M1-0000-01', '1MYPROD-V7R4M1-0000-02', and '1MYPROD-V7R4M1-0000-03'.
2. Within the appropriately named directory just created, put the correct document(s). All translated

versions of document 1 (from the example these are 1MYPROD-V7R4M1-0000-01_en, 1MYPROD-V7R4M1-0000-01_fr) will be located in the following directory: 'QIBM/UserData/LicenseDoc/1MYPROD-V7R4M1-0000-01'. Similarly, document 2 (1MYPROD-V7R4M1-0000-02_en, 1MYPROD-V7R4M1-0000-02_fr), and document 3 (1MYPROD-V7R4M1-0000-03_en, 1MYPROD-V7R4M1-0000-03_fr) will have to be located under their respectively named directories.

3. A specific language identifier must be appended to the document name for each language the document is available in. Currently there are 43 recognized language suffixes valid for software agreements. See Approved Language Suffixes for a full list.

The naming restrictions on the software agreements document name assigned to a product load include:

- Software agreement document name must be 80 or fewer characters long.
- To prevent naming conflicts with other software agreement documents, the software agreement documents must be uniquely named. One suggestion for uniquely naming your software agreement documents would be:
    - Insert the Product ID - Version/Release/Modification - Option information for the product load into the document name. Example:
        - Product: 1MYPROD, Version/Release/Modification: V7R4M1, Option: 0002
        - Software agreement document name: 1MYPROD-V7R4M1-0002
- Software agreement documents must be located in a directory with the same name as the document in the '/QIBM/UserData/LicenseDoc' directory.
- Software agreement documents must have a language identifier appended to the document name. Note: The directory name in which the document is located will not contain a language identifier, only the document name will contain the suffix.
    - From the previous example, you would create the directory '/QIBM/UserData/LicenseDoc/1MYPROD-V7R4M1-0002'. Then for each language the document will be available in, create a file named '1MYPROD-V7R4M1-0002_nn' (where '_nn' signifies the language of this document) in this directory. See Approved Language Suffixes for a full list of recognized languages.
- Software agreement documents must be stored in UTF 16 (Big Endian). See the following article for more detail on creating and storing your software agreements in UTF 16: Generate online software agreements.

Approved Language Suffixes

The following table displays the language suffixes recognized in V5R2M0 that may be used to identify the translated software agreement documents for this product load.

Language	Suffix	Language	Suffix
Albanian	_sq	Arabic	_ar
Bulgarian	_bg	Byelorussian	_be
Catalan	_ca	Croatian	_hr
Czech	_cs	Danish	_da
Dutch	_nl	English	_en

Estonian	_et	Finnish	_fi
French	_fr	German	_de
Greek	_el	Hebrew	_iw
Hindu	_hi	Hungarian	_hu
Icelandic	_is	Italian	_it
Japanese	_ja	Korean	_ko
Laotion	_lo	Latvian	_lv
Lithuanian	_lt	Macedonian	_mk
Polish	_pl	Norwegian	_no
Portugese	_pt	Portugese	_pt_BR
Romanian	_ro	Russian	_ru
Serbian	_sr	Simplified Chinese	_zh_CN
Slovakian	_sk	Slovenian	_sl
Spanish	_es	Swedish	_sv
Thai	_th	Traditional Chinese	_zh_TW
Turkish	_tr	Ukranian	_uk
Vietnamese	_vi		

Error Messages

Message ID	Error Message Text
CPF0CA1 E	No language load defined.
CPF0CA2 E	Code load ID &9 not valid.
CPF0CA3 E	Code load &4 supported.
CPF0CB1 E	Registration identifier not valid.
CPF0CB2 E	Product identifier &1 not valid.
CPF0CB3 E	Value for reserved field not valid.
CPF0C1B E	Requirements between parameters not satisfied.

CPF0C16 E Object &1 type &3 already exists in library &2.

CPF0C17 E *&3 object already exists for product &4 release &5.

CPF0C18 E Registration identifier &7 not valid for product &4 release &5.

CPF0C19 E Damage occurred on object &1 in library &2.

CPF0C4A E Product record not found.

CPF0C4B E Product availability object &2/&1 recovery required.

CPF0C4C E Cannot allocate object &1 in library &2.

CPF0C4D E Error occurred while processing object &1 in library &2.

CPF0C5B E Duplicate primary product directory.

CPF0C5C E Specified product directory name not allowed.

CPF0C5D E Product directory not allowed.

CPF0C5E E Too many product directories specified.

CPF0C5F E Product directory is not in valid format.

CPF0C54 E Data in product record not correct.

CPF0C55 E Registration ID problem with path.

CPF0C58 E Cannot add duplicate path name to *PRDLOD.

CPF0C59 E Directory in use.

CPF0C8A E Product option &1 not valid.

CPF0C8B E Product load type &1 not valid.

CPF0C8C E Number of additional libraries not valid.

CPF0C8D E Preoperation exit program information not valid.

CPF0C8E E Preoperation exit program library not valid.

CPF0C8F E Number of folders not valid.

CPF0C81 E Product load &6 in library &5 not created.

CPF0C82 E Error occurred while creating product load &6 in library &5.

CPF0C83 E Previous level folder not specified.

CPF0C84 E Load identifier &4 not valid.

CPF0C85 E Duplicate library &5 specified.

CPF0C87 E Library &1 not allowed.

CPF0C9B E Authority &1 not valid.

CPF0C9C E	Secondary language library name required.
CPF0C9D E	Minimum target release not valid.
CPF0C91 E	Code load does not exist.
CPF0C92 E	Folder name not correct.
CPF0C93 E	More than one root folder specified.
CPF0C94 E	Object name *LNG not valid for code load.
CPF0C95 E	*CODE not valid for library.
CPF0C96 E	Secondary language library not valid.
CPF0C97 E	Duplicate folder &5 in folder list.
CPF0C98 E	Additional development library &5 not found in code load.
CPF0C99 E	Product definition object not found.
» CPF0D11 E	Software agreement enablement only valid for loads of type *CODE.
CPF0D12 E	Number of software agreement documents specified is not valid.
CPF0D13 E	Software agreement document name not valid.
CPF0D14 E	Software agreements enablement only valid with minimum target release values after V5R2M0.«
CPF0613 E	User profile does not have enough storage assigned.
CPF24B4 E	Severe error while addressing parameter list.
CPF3CF1 E	Error code parameter not valid.
CPF3C21 E	Format name &1 is not valid.
CPF3C29 E	Object name &1 is not valid.
CPF3C90 E	Literal value cannot be changed.
CPF358A E	Release not valid.
CPF9810 E	Library &1 not found.
CPF9818 E	Object &2 in library &3 not created.
CPF9819 E	Object &2 in library &3 not created.
CPF9820 E	Not authorized to use library &1.
CPF9830 E	Cannot assign library &1.
CPF9872 E	Program or service program &1 in library &2 ended. Reason code &3.

API Introduced: V2R3

Create Program Temporary Fix (QPZCRTFX) API

Required Parameter Group:

1	PTF information	Input	Char(50)
2	Development library name	Input	Char(10)
3	Objects	Input	Array (*) of Char (20)
4	Number of objects	Input	Binary(4)
5	Documents	Input	Array (*) of Char(73)
6	Number of documents	Input	Binary(4)
7	Requisite PTFs	Input	Array(*) of Char(24)
8	Number of requisite PTFs	Input	Binary(4)
9	Exit programs	Input	Array(*) of Char(84)
10	Number of exit programs	Input	Binary(4)
11	Problem IDs	Input	Array(*) of Char(10)
12	Number of problem IDs	Input	Binary(4)
13	Cover letters	Input	Array(*) of Char(44)
14	Number of cover letters	Input	Binary(4)
15	Error code	I/O	Char(*)

Optional Parameter Group 1:

16	Directory information	Input	Char(*)
17	Number of directories	Input	Binary(4)

Optional Parameter Group 2:

18	Additional parameter information	Input	Char(*)
19	Additional parameter information format name	Input	Char(8)

Default Public Authority: *EXCLUDE

Threadsafe: No

This API should only be used by organizations to create program temporary fixes (PTFs) for products that they develop.

The Create Program Temporary Fix (QPZCRTFX) API creates a PTF save file and optionally creates cover letters in the general purpose library (QGPL). The save file contains a PTF control object and any number of fix objects. The save file name is the PTF identifier preceded by the letter Q. If a file with the same name already

exists in QGPL, a unique name is generated by the system. This name is a timestamp preceded by the letter Q. After creating the PTF, you can use the Display PTF (DSPPTF) command to view the PTF attributes.

PTFs can only be created for products that are installed.

PTFs must be created by a profile that is known to exist on all systems. This allows a PTF to be loaded on any system that has the product installed.

Authorities and Locks

API Public Authority

*EXCLUDE

Create library (CRTLIB) command

*USE authority to the command and all authorities required by the command.

Create save file (CRTSAVF) command

*USE authority to the command and all authorities required by the command.

The following authorities are required when specifying the indicated input parameter:

Cover Letter Parameter

*USE authority to input cover letter file

*EXECUTE authority to input cover letter library

*OBJOPR, *OBJMGR, *ADD, *DLT authority to the QAPZCOVER file in library QGPL

Object Parameter

*CHANGE authority to the object

*EXECUTE authority to the development library

Exit program parameter

*CHANGE authority to the exit program

*EXECUTE authority to the exit program library

Document parameter

*USE authority to the SAVDLO command and all authorities required by the command.

Directory information parameter

*USE authority to the SAV command and all authorities required by the command.

*USE authority to the CPY command and all authorities required by the command.

*USE authority to the CRTDIR command and all authorities required by the command.

Required Parameter Group

PTF information

INPUT; CHAR(50)

Attributes of the PTF to be created. See PTF Information Format for more information about this field.

Development library name

INPUT; CHAR(10)

The library in which the fix is located. This can be any library.

Objects

INPUT; ARRAY(*) of CHAR(20)

The name and type of each object to be included in the PTF. The first 10 characters contain the name, and the second 10 characters contain the external type of the object.

Object name	The name of the object.
Object type	The external type of the object. This must be preceded by an asterisk (*). For more information, refer to the System Manager Use book.

Number of objects

INPUT; BINARY(4)

The number of objects listed in the objects parameter. This number must be in the range of 1 through 300.

Documents

INPUT; ARRAY(*) of CHAR(73)

The name of the documents that are to be included in the PTF.

The create PTF function copies the document from a subfolder using the name specified, followed by /QP. For example, if a PTF is being created for a product folder called PRODUCT, the fix objects must be developed in a subfolder named PRODUCT/QP. The document is installed into the product folder PRODUCT during the apply PTF operation. The QP subfolder allows you to develop a PTF without changing the product.

Document name	The name of the document including the path name.

Number of documents

INPUT; BINARY(4)

The number of documents listed in the documents parameter. This number must be in the range of 1 through 300.

Requisite PTFs

INPUT; ARRAY(*) of CHAR(24)

The list of requisite PTFs. A **requisite relationship** exists when one PTF requires that another PTF also

be applied. It is a **prerequisite relationship** if the other PTF does not require the first. It is a **corequisite relationship** if the other PTF does require the first. Prerequisite PTFs must exist within the same product. A prerequisite PTF must already exist on the system or the create operation will fail. Corequisite PTFs must exist within the same product, option, load id, and release.

For more information on this structure, see Requisite PTF Format.

Number of requisite PTFs

INPUT; BINARY(4)

The number of PTFs listed in the requisite PTFs parameter. This number must be in the range of 1 through 300.

Exit programs

INPUT; ARRAY(*) of CHAR(84)

The PTF exit programs called when a PTF is temporarily applied, permanently applied, temporarily removed, or permanently removed. Exit programs eliminate the need for you to manually carry out special instructions to install the PTF. The run option field of this parameter determines when the exit program is called.

Shipping the same exit program in two PTFs causes one PTF to supersede the other.

For more information on this structure, see Exit Programs Format and Program Temporary Fix Exit Program.

Number of exit programs

INPUT; BINARY(4)

The number of exit programs listed in the exit programs parameter. This number must be in the range of 1 through 50.

Problem IDs

INPUT; ARRAY(*) of CHAR(10)

A list of the problem IDs for problems that this PTF fixes. By listing the problem IDs, the symptom strings associated with those problems will be included in the PTFs.

Number of problem IDs

INPUT; BINARY(4)

The number of problem IDs listed in the problem IDs parameter. This number must be in the range of 1 through 300.

Cover letters

INPUT; ARRAY(*) of CHAR(44)

A cover letter can be created for each of the national language versions (NLV) that IBM supports. A member that contains source for each PTF cover letter must be supplied as input to the API. The cover letter file can be a source file with a maximum record length of 92 or a physical file with record length of 80. The cover letter must be in the file before this API is called. Only one cover letter per NLV is allowed.

For more information on this structure see Cover Letter Format.

Number of cover letters

INPUT; BINARY(4)

The number of cover letters listed in the cover letter parameter. This number must be in the range of 1 through 50.

Error code

I/O; CHAR(*)

The structure in which to return error information. For the format of the structure, see Error Code Parameter.

Optional Parameter Group 1

Directory information

INPUT; Array(*) of CHAR(*)

Identifies the information for directory objects that are included in the PTF. See Directory Information Format for more information about this field.

Number of directories

INPUT; BINARY(4)

The number of directories listed in the directory information parameter. This number must be in the range of 0 through 30.

Optional Parameter Group 2

Additional parameter information

INPUT; CHAR(*)

The additional information to use when creating this PTF.

Additional parameter information format name

INPUT; CHAR(8)

The format of the data specified in the additional parameter information. The possible format name is:

PTFC0100 The format contains information to create a PTF that contains job preconditions and object preconditions. For details, see PTFC0100 Format.

PTF Information Format

For detailed descriptions of each field, see Field Descriptions.

Offset		Type	Field
Dec	Hex		
0	0	CHAR(7)	PTF ID
7	7	CHAR(7)	Product ID
14	E	CHAR(6)	Release level

20	14	CHAR(4)	Product option
24	18	CHAR(10)	Primary object library name
34	22	CHAR(4)	Load ID
38	26	CHAR(6)	OS/400 target release
44	2C	CHAR(6)	Reserved

Requisite PTF Format

Each entry in the array for the requisite PTFs parameter has the following format. For detailed descriptions of each field, see Field Descriptions.

Offset			
Dec	**Hex**	**Type**	**Field**
0	0	CHAR(7)	PTF ID
7	7	CHAR(16)	Reserved
23	17	CHAR(1)	Requisite type

Exit Programs Format

Each entry in the array for the exit programs parameter has the following format. For detailed descriptions of each field, see Field Descriptions.

Offset			
Dec	**Hex**	**Type**	**Field**
		CHAR(10)	Exit program name
		CHAR(10)	Exit program library name
		CHAR(7)	Run option
		CHAR(7)	Exit program type
		CHAR(50)	User data

Cover Letter Format

Each entry in the array for the cover letter parameter has the following format. The information must be presented in the order listed below. The exact offsets for each entry are not given. For detailed descriptions of each field, see Field Descriptions.

Offset			
Dec	**Hex**	**Type**	**Field**
		CHAR(10)	Cover letter file name

		CHAR(10)	Cover letter library name
		CHAR(10)	Cover letter member name
		CHAR(4)	NLV
		CHAR(10)	Reserved

Directory Information Format

Each record in the array for the directory information parameter has the following format. The information must be presented in the order listed below. The exact offsets for each entry are not given. For detailed descriptions of each field, see Field Descriptions.

The following restrictions exist when you are assigning directory names:

- You cannot specify /QSYS.LIB or /QDLS directories.
- You must specify unique path names.
- Do not begin or end the path name with a forward slash.
- Do not use a blank in the directory path name.
- Do not use any of the character combinations of "." or ".." in the directory path name.

Offset		Type	Field
Dec	Hex		
		Array of CHAR(*)	Directory information record
		BINARY(4)	Offset to the next directory information record
		BINARY(4)	Offset to the development directory name
		BINARY(4)	Length of the development directory name
		BINARY(4)	Offset to the product directory name
		BINARY(4)	Length of the product directory name
		BINARY(4)	Offset to the first directory object information record
		BINARY(4)	Number of directory objects
		CHAR(*)	Development directory name
		CHAR(*)	Product directory name

Directory Object Information Record Format

Each record in the array for the directory object information parameter has the following format. The information must be presented in the order listed below. The exact offsets for each entry are not given. For detailed descriptions of each field, see Field Descriptions.

Offset		Type	Field
Dec	Hex		

		Array of CHAR(*)	Directory object information record
		BINARY(4)	Length of directory object name
		BINARY(4)	Displacement to next directory object information record
		CHAR(*)	Directory object name

PTFC0100 Format

This information defines the format for the additional parameter information when the additional parameter information format name is PTFC0100. For detailed descriptions of each field, see Field Descriptions.

Offset		Type	Field
Dec	Hex		
0	0	BINARY(4)	Offset to first job precondition record
4	4	BINARY(4)	Number of job precondition records
8	8	BINARY(4)	Length of each job precondition record
12	C	BINARY(4)	Offset to first object precondition record
16	10	BINARY(4)	Number of object precondition records
20	14	BINARY(4)	Length of each object precondition record

Job Precondition Record

An array containing the names of the jobs or subsystems that must not be active when the PTF is being immediately applied or removed temporarily. When this PTF is applied or removed and this job or subsystem is active, the PTF will not be allowed to be processed.

Note: PTF processing will not prevent the job or subsystem from becoming active after this check is made, but before the PTF is actually processed.

Each record in the job preconditions array has the following format. The information must be presented in the order listed below. The exact offsets for each entry are not given. For detailed descriptions of each field, see Field Descriptions.

Offset		Type	Field
Dec	Hex		
		CHAR(1)	Precondition type
		CHAR(10)	Precondition name
		CHAR(*)	Reserved

Object Precondition Record

An array containing the names of the objects that must not be allocated when the PTF is being immediately applied or removed temporarily. When this PTF is applied or removed and this object has active or waiting locks, the PTF will not be allowed to be processed.

Note: PTF processing will not prevent the object from being allocated after this check is made, but before the PTF is actually processed.

Each entry in the object preconditions array has the following format. The information must be presented in the order listed below. The exact offsets for each entry are not given. For detailed descriptions of each field, see Field Descriptions.

Offset		Type	Field
Dec	Hex		
		CHAR(10)	Precondition object name
		CHAR(10)	Precondition library name
		CHAR(10)	Precondition object type
		CHAR(*)	Reserved

Field Descriptions

Cover letter file name. The name of the file where the cover letter can be located.

Cover letter library name. The name of the library where the cover letter file can be located.

Cover letter member name. The member name that contains the cover letter.

Development directory name. The name of the directory where the directory objects that will be in the PTF reside. The possible special value is:

**PRDDIR* The development directory name is the same as the product directory name.

Directory information record. The information about each directory for this PTF.

Directory object information record. The information about the objects for this directory.

Directory object name. The name of the directory object to include in the PTF.

Exit program library name. The library where the exit program can be found. If the exit program is part of the PTF, this is the library in which it exists currently. If the exit program is part of the product, this is the primary library where the exit program exists.

Exit program name. The name of the exit program.

Exit program type. Whether the exit program is to be included in this PTF. The possible values are:

**PTF* The exit program is to be included in the PTF. The exit program must exist in the library specified in the exit program library field.

**OBJLST* The exit program is part of the product and should not be included in the PTF. The exit program must exist in one of the following:

- The object list for the product, option, and load of the PTF being created.
- The principal library for the base option (*BASE) of the product.

Length of each job precondition record. The length of the job precondition record. The length must be set to 11.

Length of each object precondition record. The length of the object precondition record. The length must be set to 30.

Length of the development directory name. The length of the development directory name. The length of a development directory cannot exceed 240 characters.

Length of the directory object name. The length of the directory object name. The length of a directory object exceed 255 characters.

Length of the product directory name. The length of the product directory name. The length of a product directory cannot exceed 240 characters.

Load ID. The load ID of the product load for the PTF. This will be a language load if the PTF is for textual data, or it will be the code load.

Number of directory objects. The number of objects that exist in the array of directory object information records. This number must be in the range of 1 through 100.

Number of job precondition records. The number of job preconditions that exist. This number must be in the range of 0 through 10.

Number of object precondition records. The number of object preconditions that exist. This number must be in the range of 0 through 10.

NLV. The NLV of the cover letter. This must be a valid system NLV.

Offset to the development directory name. The byte offset from the beginning of the directory information parameter to the beginning of the name of the development directory.

Offset to the first directory object information record. The byte offset from the beginning of the directory information parameter to the first directory object information record.

Offset to the first job precondition record. The byte offset from the beginning of the additional information parameter to the beginning of the first job precondition record.

Offset to the first object precondition record. The byte offset from the beginning of the additional information parameter to the beginning of the first object precondition record.

Offset to the next directory information record. The byte offset from the beginning of the directory information parameter to the beginning of the next directory information record.

Offset to the product directory name. The byte offset from the beginning of the directory information parameter to the beginning of the name of the product directory.

OS/400 target release. The earliest release of the operating system on which you intend to load and apply the PTF. This must be left-justified. If this is blank, the current release is assumed. The possible special values follow:

**CUR* The PTF is to be loaded, applied to, and used on the release of the operating system currently running on your system. The PTF also can be applied on a system with any later release of the operating system installed.

**PRV* The PTF is to be loaded, applied to, and used on the previous release with modification level 0 of the operating system. The PTF also can be applied on a system with any later release of the operating system installed.

Target release The release of the operating system on which you intend to load and apply the PTF. The release level is specified in the format V*x*R*y*M*z*, where V*x* is the version, R*y* is the release, and M*z* is the modification level. Valid values depend on the current version, release, and modification level, and they change with each new release.

Note: This PTF can be loaded and applied on any release after the specified target release.

Precondition library name. The name of the library where the object specified in the precondition object name field resides.

Precondition name. The name of the job or subsystem that must not be active when this PTF is temporarily applied or removed immediately. A specific name or a generic name can be specified. This field must be blanks when the precondition type is 3.

Precondition object name. The name of the object that must not be allocated when this PTF is applied temporarily or removed immediately. A specific name or a generic name may be specified.

Precondition object type. The type of the object specified in the precondition object name field.

Precondition type. The type of the precondition specified in the precondition name. The possible values are:

1 The precondition name indicates a job.

2 The precondition name indicates a subsystem.

3 The system must be in restricted state for this PTF to be immediately applied or removed. The precondition name field must be blanks when this value is specified.

Primary object library name. The library in which the objects are to be placed when the PTF is applied. If necessary, the PTF apply operation maps the primary library that is specified when the PTF was created in the actual installed library. Two cases where this is important are:

- The product load has been installed into a secondary language library.
- Dynamic library renaming was used when the product was installed.

Product directory name. The name of the directory defined by the product that is the default directory where the objects will reside when the PTF is applied. The length of a product directory cannot exceed 240 characters.

Product ID. The product for which the PTF is being created. This product must be installed on the system.

Product option. The option of the product for which the PTF is being created. All objects in the PTF must be for the same option and the same library within the option.

PTF ID. The ID by which the PTF is to be known. The identifier must be 7 characters. The first character must be numeric. The second and third characters must be alphabetic. The same identifier can be used only once for each product and release level.

Release level. The version, release, and modification level of the product in the format V*x*R*y*M*z*. Valid values

for *x* and *y* are 0 through 9, and valid values for *z* are 0 through 9 or A through Z.

Requisite type. The type of requisite relationship. If this is blank, a prerequisite relationship is assumed. The possible values are:

1 The requisite PTF is a prerequisite of this PTF. The requisite PTF is required by this PTF, but it does not require this PTF. It cannot specify this PTF as a prerequisite. It must be applied before or with this PTF. If it is applied with this PTF, the system applies it first.

2 The requisite PTF is a corequisite of this PTF; it is required by this PTF, and it requires this PTF. It must specify this PTF as a corequisite. Corequisite PTFs must be applied together. When they are applied together, the system may apply either of them first. This value is not valid for PTFs with a target release earlier than V4R2M0.

(blank) A value of 1 is assumed. The requisite PTF is a prerequisite of this PTF. The requisite PTF is required by this PTF, but it does not require this PTF.

Reserved. An error will be signaled if this field does not contain blanks.

Run option. When the exit program is to be run. The possible values are:

**BOTH* The exit program will be run at the end of apply and remove processing.

**APPLY* The exit program will be run at the end of apply processing.

**REMOVE* The exit program will be run at the end of remove processing.

**PREAPY* The exit program will be run before the PTF is applied and at the end of apply processing.

**PRERMV* The exit program will be run before the PTF is removed and at the end of remove processing.

**PREBTH* The exit program will be run before the PTF is removed and at the end of remove processing. It is also run before the PTF is applied and at the end of apply processing.

User data. Any data you want to pass to the exit program.

Error Messages

Message ID	Error Message Text
CPF24B4 E	Severe error while addressing parameter list.
CPF3CF1 E	Error code parameter not valid.
CPF3C29 E	Object name &1 is not valid.
CPF3C31 E	Object type &1 is not valid.
CPF3C90 E	Literal value cannot be changed.
CPF35BC E	Object type &1 not supported.
CPF35CC E	Library required for building PTFs already exists.

CPF35DA E Folder &1 not valid.

CPF35DB E Duplicate documents specified.

CPF35DC E Primary library not found.

CPF35DD E Problem &1 does not exist.

CPF35DF E Value for target release not valid.

CPF35D3 E Cover letter not copied.

CPF35D4 E Cover letter file record length too long.

CPF35D5 E Cover letter NLV not valid.

CPF35D6 E Duplicate exit programs specified.

CPF35D8 E Exit program &1 not valid.

CPF35D9 E Duplicate objects specified.

CPF3505 E Corequisite PTF &1-&2 &3 contains common objects.

CPF3507 E Corequisite PTF &1-&2 &3 not specified.

CPF3509 E Specified corequisite PTF &1-&2 &3 not valid.

CPF357A E Parameter value not valid.

CPF357B E Product not found.

CPF357D E Document or folder name not correct.

CPF3570 E No PTF IDs available in range.

CPF3571 E PTF ID &1 not within valid range.

CPF3572 E PTF &2-&1 &3 already exists.

CPF3573 E Resources required for product &1 are not available.

CPF3574 E PTF ID not valid.

CPF358A E Release not valid.

CPF358B E PTF not created.

CPF358C E Create PTF not allowed for product &1.

CPF358D E Run option not valid.

CPF358E E Exit program type not valid.

CPF359C E Requisite type not valid.

CPF35EC E Duplicate requisites specified.

CPF3901 E PTF &1-&2 &3 not created.

CPF9872 E	Program or service program &1 in library &2 ended. Reason code &3.

API introduced: V2R3

Top | Software Product APIs | APIs by category

Create PTF Group (QpzCreatePtfGroup) API

Required Parameter Group:

1	PTF group information	Input	CHAR(*)
2	PTF group information format name	Input	CHAR(8)
3	Input variable	Input	CHAR(*)
4	Input variable format name	Input	CHAR(8)
5	Related PTF groups	Input	ARRAY(*) of CHAR(60)
6	Number of related PTF groups	Input	BINARY(4)
7	CCSID	Input	BINARY(4)
8	Error code	I/O	CHAR(*)

Service Program Name: QPZGROUP

Default Public Authority: *USE

Threadsafe: No

The Create PTF Group (QpzCreatePtfGroup) API creates a PTF group.

A PTF group consists of a number of program temporary fixes (PTFs) for the purpose of managing those PTFs as one entity. After creating the PTF group you can use the Work with PTF Groups (WRKPTFGRP) command or the List PTF Group Details (QpzListPtfGroupDetails) API to view the PTF group. PTF groups can also be managed using Management Central.

Authorities and Locks

Work with PTF Groups (WRKPTFGRP) command

*USE

File or User Space Authority

*USE

File or User Space Library Authority

*EXECUTE

File or User Space Lock

*EXCLRD

Lock conflicts may occur if this API is called while another PTF or PTF group operation is in progress.

Required Parameter Group

PTF group information

INPUT; CHAR(*)

Attributes of the PTF group to be created. The format of this information is described by the PTF group information format name parameter.

PTF group information format name

INPUT; CHAR(8)

The format of the information in the PTF group information parameter. The possible format names are:

GRPC0100 For details, see GRPC0100 Format.

Input variable

INPUT; CHAR(*)

The name of the user space or file that contains the list of PTFs to be included in this PTF group. The format of the name is defined by the input variable format name parameter.

For format GRPI0000, the input variable parameter should be specified as a CHAR(10) field with a value of *NONE, or the parameter should be a null pointer.

For format GRPI0100, the input variable parameter will contain the qualified user space name in the following format:

Offset		Type	Field
Dec	Hex		
0	0	CHAR(10)	User space name
10	A	CHAR(10)	Library name

For format GRPI0200, the input variable parameter will contain the physical file member name in the following format:

Offset		Type	Field
Dec	Hex		
0	0	CHAR(10)	File name
10	A	CHAR(10)	Library name
20	14	CHAR(10)	File member name

Input variable format name

INPUT; CHAR(8)

The format of the information in the input variable parameter. The possible format names are:

GRPI0000 There are no PTFs to be included in this PTF group. You must include at least one related PTF group in order to create this PTF group.

GRPI0100 The input variable parameter defines the name and format of a user space that contains the list of PTFs to be included in this PTF group. For details, see GRPI0100 Format.

GRPI0200 The input variable parameter defines the name and format of a physical file member that contains the list of PTFs to be included in this PTF group. For details, see GRPI0200 Format.

Related PTF groups

INPUT; ARRAY(*) of CHAR(60)

The name of the PTF groups that are related to this PTF group. Related PTF groups are included by specifying the PTF group name only, not by level. Related PTF groups are used when determining the overall status of this PTF group and are also included when the PTF group is distributed and installed using Management Central.

Number of related PTF groups

INPUT; BINARY(4)

The number of related PTF groups listed in the related PTF groups parameter. This number must be in the range of 0 through 300.

CCSID.

INPUT; BINARY(4)

The coded character set ID for the PTF group name, description, and related PTF group names. Valid values are 0 through 65533. If a value of 0 is specified, the names and description are assumed to be in the CCSID of the job.

Error code

I/O; CHAR(*)

The structure in which to return error information. For the format of the structure, see Error Code Parameter.

GRPC0100

Describes the format of the information in the PTF group information parameter.

For detailed descriptions of each field, see the Field Descriptions.

Offset		Type	Field
Dec	Hex		
0	0	BINARY(4)	Length of PTF group information
4	4	CHAR(60)	PTF group name
64	40	CHAR(100)	PTF group description
164	A4	BINARY(4)	PTF group level
168	A8	CHAR(1)	Replace group

GRPI0100 format

Describes the format of the PTF List in the user space specified in the input variable parameter.

For detailed descriptions of each field, see the Field Descriptions.

This format of the data that exists in the user space must be in the following format.

Offset		Type	Field
Dec	Hex		
0	0	BINARY(4)	Offset to the first PTF record
4	4	BINARY(4)	Number of PTF records
8	8	BINARY(4)	Length of each PTF record
Note: The following fields are repeated for each PTF in the list.			
		CHAR(7)	PTF ID
		CHAR(7)	Product ID
		CHAR(6)	Release
		CHAR(4)	Product option
		CHAR(4)	Product load ID
		CHAR(2)	Minimum level
		CHAR(2)	Maximum level

GRPI0200 format

Defines the format of the PTF List in the physical file member specified in the input variable parameter. The data in this file must be in the format defined by file QADSPPTF in library QSYS. This file format is used when specifing OUTPUT(*OUTFILE) on the Display PTF (DSPPTF) command or by using the Create PTF Package(CRTPTFPKG) command in the System Manager/400 licensed product.

The following fields in this file are required to contain valid information:

- SCPPID
- Product ID
- SCPTFID
- PTF ID
- SCPTFV
- Release
- SCOPTP
- Product option
- SCENLG
- Product load ID. A special value of "CODE" indicates the PTF is for a feature type of *CODE.
- SCMNLV
- Minimum level.

- SCMXLV
- Maximum level.

Field Descriptions

File member name. The name of the physical file member that contains the list of PTFs to be included in this PTF group. The data in this member must be in the format defined by file QADSPPTF in library QSYS.

File name. The name of the physical file that contains the list of PTFs to be included in this PTF group.

Library name. The name of the library where the user space or physical file can be located. You can use these special values for the library name:

**CURLIB* The job's current library.

**LIBL* The library list.

Length of each PTF record. The length of the data in each PTF record in the user space. The only valid value is 32.

Length of PTF group information. The length of the data in the PTF group information format, including this field. The only valid value is 169.

Maximum level. The indicator of the highest level of the product on which this PTF can be installed. If the minimum and maximum levels are the same, then this PTF can only be installed on one level of the product. The level can be AA to 99. If the PTF has no maximum level, this field should be blanks.

Minimum level. The indicator of the lowest level of the product on which this PTF can be installed. If the minimum and maximum levels are the same, then this PTF can only be installed on one level of the product. The level can be AA to 99. If the PTF has no minimum level, this field should be blanks.

Number of PTF records. The number of PTF records that exist in the user space.

Offset to the first PTF record. The byte offset from the beginning of the user space to the first PTF record.

Product ID. The product identifier of the PTF.

Product load ID. The load ID of the product load for the PTF. A special value of "CODE" indicates the PTF is for a feature type of *CODE. For language loads, you must specify a valid national language version.

Product option. The option of the product for the PTF. Valid values are 0000 through 0099.

PTF ID. The identifier of the PTF to be included within the PTF group.

PTF group description. The text description of the PTF group. The description must be able to be contained in a 50-byte EBCDIC field. For example, 50 single-byte characters or 24 double-byte characters with one shift-out character and one shift-in character.

PTF group level. The level of the PTF group. The level can be in the range 1 through 99999. When creating a different version of a PTF group, the new level specified should be higher. A higher level is considered to be a more recent version of the PTF group.

PTF group name. The name of the PTF group you are creating. The first character must be numeric in the range 0 through 9. The remaining characters cannot contain imbedded blanks or an asterisk(*). The name must

be able to be contained in a 30-byte EBCDIC field. For example, a name with single-byte characters must be a maximum of 30 characters and be padded with blanks. For names with double-byte characters, the name must begin with a numeric character followed by one shift-out character, then a maximum of 13 double-byte characters followed by one shift-in character.

Release. The version, release, and modification of the PTF in the format V*x*R*y*M*z*. Valid values for *x* and *y* are 0 through 9, and valid values for *z* are 0 through 9 or A through Z.

Replace group. The action to take when a PTF group of the same name already exists on the system. Only one level of a PTF group can exist on the system. Valid values for this field are:

0 Do not replace an existing PTF group of the same name, regardless of the level of the PTF group.

1 Replace an existing PTF group of the same name only when the of the level of the PTF group being created is higher than the level of the existing PTF group.

2 Replace an existing PTF group of the same name only when the of the level of the PTF group being created is equal to or higher than the level of the existing PTF group.

3 Always replace an existing PTF group of the same name.

User space name. The name of the user space that contains the list of PTFs to be included in this PTF group.

Error Messages

Message ID	Error Message Text
CPF0CEE E	Unable to convert data to CCSID &1.
CPF24B4 E	Severe error while addressing parameter list.
CPF3598 E	PTF function already in progress.
CPF36AE E	Duplicate related PTF group &1 not allowed.
CPF36AF E	PTF group function already in progress.
CPF36A0 E	Value for PTF group level &1 not valid.
CPF36A1 E	No PTFs or related PTF groups specified.
CPF36A2 E	Length of PTF group name or text too long.
CPF36A3 E	PTF group already exists.
CPF36A6 E	PTF group name &1 not valid.
CPF36A7 E	PTF data for PTF group is not valid.
CPF36B0 E	Field at offset &1 in user space &2 not valid.
CPF36B1 E	Value for parameter &1 not valid.
CPF3BC7 E	CCSID &1 outside of valid range.
CPF3CF1 E	Error code parameter not valid.

CPF3CF2 E	Error(s) occurred during running of &1 API.
CPF3C21 E	Format name &1 is not valid.
CPF3C3C E	Value for parameter &1 not valid.
CPF9801 E	Object &2 in library &3 not found.
CPF9802 E	Not authorized to object &2 in &3.
CPF9803 E	Cannot allocate object &2 in library &3.
CPF9820 E	Not authorized to use library &1.
CPF9848 E	Cannot open file &1 in library &2 member &3.
CPF9872 E	Program or service program &1 in library &2 ended. Reason code &3.

API introduced: V5R2

Top | Software Product APIs | APIs by category

Delete Product Definition (QSZDLTPD) API

Required Parameter Group:

1	Qualified product definition name	Input	Char(20)
2	Error code	I/O	Char(*)

Default Public Authority: *USE

Threadsafe: No

The Delete Product Definition (QSZDLTPD) API deletes a single product definition object.

Authorities and Locks

Library Authority

*USE

Library Lock

*SHRUPD

Object Lock

*EXCL

Product Definition Authority

*OBJEXIST

Required Parameter Group

Qualified product definition name

INPUT; CHAR(20)

The first 10 characters contain the product definition object name. The second 10 characters contain the name of the library where the product definition object is located.

Error code

I/O; CHAR(*)

The structure in which to return error information. For the format of the structure, see Error Code Parameter.

Error Messages

Message ID	Error Message Text
CPF0C4B E	Product availability object &2/&1 recovery required.
CPF0C52 E	Product definition &3 in library &4 not deleted.
CPF2105 E	Object &1 in &2 type *&3 not found.
CPF2110 E	Library &1 not found.
CPF2113 E	Cannot allocate library &1.
CPF2114 E	Cannot allocate object &1 in &2 type *&3.
CPF2125 E	No objects deleted.
CPF2176 E	Library &1 damaged.
CPF2182 E	Not authorized to library &1.
CPF2189 E	Not authorized to object &1 in &2 type *&3.
CPF24B4 E	Severe error while addressing parameter list.
CPF3CF1 E	Error code parameter not valid.
CPF3C29 E	Object name &1 is not valid.
CPF3C90 E	Literal value cannot be changed.
CPF9872 E	Program or service program &1 in library &2 ended. Reason code &3.

API Introduced: V2R3

Delete Product Load (QSZDLTPL) API

Required Parameter Group:

1	Qualified product load name	Input	Char(20)
2	Error code	I/O	Char(*)

Default Public Authority: *USE

Threadsafe: No

The Delete Product Load (QSZDLTPL) API deletes a single product load object.

Authorities and Locks

Library Authority

*USE

Library Lock

*SHRUPD

Object Lock

*EXCL

Product Load Authority

*OBJEXIST

Required Parameter Group

Qualified product load name

INPUT; CHAR(20)

The first 10 characters contain the product load object name. The second 10 characters contain the name of the library where the product load object is located.

Error code

I/O; CHAR(*)

The structure in which to return error information. For the format of the structure, see Error Code Parameter.

Error Messages

Message ID	Error Message Text
CPF2105 E	Object &1 in &2 type *&3 not found.
CPF2110 E	Library &1 not found.
CPF2113 E	Cannot allocate library &1.
CPF2114 E	Cannot allocate object &1 in &2 type *&3.
CPF2125 E	No objects deleted.
CPF2176 E	Library &1 damaged.
CPF2182 E	Not authorized to library &1.
CPF2189 E	Not authorized to object &1 in &2 type *&3.
CPF24B4 E	Severe error while addressing parameter list.
CPF3CF1 E	Error code parameter not valid.
CPF3C29 E	Object name &1 is not valid.
CPF3C90 E	Literal value cannot be changed.
CPF9872 E	Program or service program &1 in library &2 ended. Reason code &3.

API Introduced: V2R3

Delete PTF Group (QpzDeletePtfGroup) API

Required Parameter Group:

1	PTF group name	Input	CHAR(60)
2	PTF group level	Input	BINARY(4)
3	Delete related PTF groups	Input	CHAR(1)
4	CCSID	Input	BINARY(4)
5	Error code	I/O	CHAR(*)

Service Program Name: QPZGROUP

Default Public Authority: *USE

Threadsafe: No

The Delete PTF Group (QpzDeletePtfGroup) API deletes a PTF group from the system. You can optionally delete all related PTF groups at the same time.

Authorities and Locks

Work with PTF Groups (WRKPTFGRP) command

*USE

Lock conflicts may occur if this API is called while another PTF or PTF group operation is in progress.

Required Parameter Group

PTF group name

INPUT; CHAR(60)

The name or generic name of the PTF group to be deleted. A generic name is a string of one or more characters followed by an asterisk (*). You can use these special values for the PTF group name:

**ALL* All PTF groups on the system are deleted.

PTF group level

INPUT; BINARY(4)

The level of the PTF group to be deleted. This parameter is ignored when a generic name or *ALL is specified for the PTF group name. Valid values for this parameter are:

0 The current level of the PTF group is deleted.

1 - 99999 The specified level of the PTF group is deleted. If the level does not exist on the system, an error will occur.

Delete related PTF groups

INPUT; CHAR(1)

Whether related PTF groups will be deleted at the time this PTF group is deleted. This parameter is ignored when *ALL is specified for the PTF group name. Valid values for this parameter are:

0 Related PTF groups will not be deleted.

1 Related PTF groups will be deleted when this PTF group is deleted. This will also delete the related PTF groups of the related PTF groups.

CCSID

INPUT; BINARY(4)

The coded character set ID for the PTF group name. Valid values are 0 through 65533. If a value of 0 is specified, the name is assumed to be in the CCSID of the job.

Error code

I/O; CHAR(*)

The structure in which to return error information. For the format of the structure, see Error Code Parameter.

Error Messages

Message ID	Error Message Text
CPF0CEE E	Unable to convert data to CCSID &1.
CPF36A2 E	Length of PTF group name or text too long.
CPF36A4 E	PTF group &1 not found.
CPF36A6 E	PTF group name &1 not valid.
CPF36AF E	PTF group function already in progress.
CPF3BC7 E	CCSID &1 outside of valid range.
CPF3C3C E	Value for parameter &1 not valid.
CPF3CF1 E	Error code parameter not valid.
CPF3CF2 E	Error(s) occurred during running of &1 API.
CPF9872 E	Program or service program &1 in library &2 ended. Reason code &3.

API introduced: V5R2

Top | Software Product APIs | APIs by category

Delete Registered Application Files (QSZDLTAF, QszDltRegAppFiles) API

Required Parameter Group:

1	Application information path name	Input	Char(*)
2	Error code	I/O	Char(*)

Service Program Name: QSZRAIRA

Default Public Authority: *EXCLUDE

Threadsafe: No

The Delete Registered Application Files (QSZDLTAF, QszDltRegAppFiles) API deletes the files listed in the files tag for the given component. This API does not delete information from the OS/400 Registered Application Information Repository. To remove the files tag from the repository, call the Update OS/400 Registered Application Information Repository API.

This API can be used to generically uninstall simple applications that only need to remove the files that were installed and need no additional processing. It also can be used as a part of a more complex uninstall operation. This API can be called to remove an application's files after the uninstall operation has performed any unique processing (such as deleting a user profile).

The user must be aware of the following:

- This API should not be used for OS/400 packaged products. The user should use the Delete License Program (DLTLICPGM) command to delete OS/400 packaged products. If a user chooses to use this API to delete objects for an OS/400 packaged product, an error message will be signalled.

- The program calling this API should not be listed in the files element of the component being processed; otherwise, locking exceptions may occur. If for some reason the program is listed, the program should be copied to a different location before calling it.

Authorities and Locks

The authorities and locks for the first parameter are:

Library Authority

*EXECUTE

Authority for user space containing XML document

*USE

User space lock

*EXCLRD

Stream file directory authority

*RX

Authority for stream file containing XML document

*R

The authorities and locks for files to delete are:

Library Authority	*OBJEXIST
Object Authority	*OBJEXIST
Stream File Directory Authority	*OBJEXIST
Stream File Object Authority	*OBJEXIST

Required Parameter Group

Application information path name

INPUT; CHAR(*)

The path name of the object that contains the XML document with the information of the components to be deleted by the API. This may be a path to a user space(*USRSPC) or a path to an stream file (*STMF). The path name should be specified in the Qlg_Path_Name_T format. If a pointer is specified in the path name format, it must be 16-byte aligned. If not, unpredictable results may occur. For more information on this structure, see Path name format. The information contained in this object must be given to the API as an XML document according to the data type definition (DTD). For a detailed description of the DTD, see Software Components DTD. For examples of how the XML input document may be structured, see XML Document when deleting component files.

Error code

I/O; CHAR(*)

The structure in which to return error information. For the format of the structure, see Error Code Parameter.

XML document when deleting component files

The XML document may be an stream file (*STMF) or a user space (*USRSPC) object. In either case, the XML document must conform to the Software Components DTD.

Even though all elements defined in the data type definition(DTD) are allowed when using this API, only the following will be taken into account and the rest will be ignored: RegAppInfoRepository, Component, Product, Version, ComponentName, Instance and Vendor. See Software Components DTD for a detailed description of each element.

The components specified should match the information in the OS/400 Registered Application Information

Repository. For example, if a component was registered without the **Version** element, the version element should be ommited when using this API.

Only files for a single component will be deleted for each component tag in the XML document.

The examples below illustrate the way to define an XML document to call this API.

Delete files for a single component.

```
<?xml ComponentVersion="1.0" encoding='ucs-2'?>
<!DOCTYPE RegAppInfoRepository SYSTEM
"/IBM/XML/DTD/RegAppInfoRepository.dtd">
<RegAppInfoRepository DTDVersion="1.0">
  <Component ProductName="My compiler" ComponentVersion="v1.1.0"
ComponentName="Tools"
             ComponentVendor="Juan Perez"> </Component>
</RegAppInfoRepository>
```

Delete files for multiple components.

Suppose you have two components which only differ in the component name('ComponentName' attribute). The following example shows how files for the two components can be deleted.

```
<?xml ComponentVersion="1.0" encoding='ucs-2'?>
<!DOCTYPE RegAppInfoRepository SYSTEM
"/IBM/XML/DTD/RegAppInfoRepository.dtd">
<RegAppInfoRepository DTDVersion="1.0">

 <Component ProductName="XYZ" ComponentVersion="v3.1.0" ComponentName="Comp
A" Instance="Unique"
      ComponentVendor="Juan Perez"> </Component>
  <Component ProductName="XYZ" ComponentVersion="v3.1.0" ComponentName="Comp
B" Instance="Unique"
      ComponentVendor="Juan Perez"> </Component>
</RegAppInfoRepository>
```

As you can see in the previous example, the difference between the two components is the 'ComponentName' attribute. An error is signalled if the document is given as follows:

```
<?xml ComponentVersion="1.0" encoding='ucs-2'?>
<!DOCTYPE RegAppInfoRepository SYSTEM
"/IBM/XML/DTD/RegAppInfoRepository.dtd">
<RegAppInfoRepository DTDVersion="1.0">
  <Component ProductName="XYZ" ComponentVersion="v3.1.0" ComponentName=""
Instance="Unique"
      ComponentVendor="Juan Perez">
  </Component>
</RegAppInfoRepository>
```

In this case, the 'ComponentName' attribute is empty; therefore, this matches two entries in the repository, so the API will not know which files should be deleted.

Error Messages

Message ID	Error Message Text
CPF24B4 E	Severe error addressing parameter list.
CPF3CF1 E	Error code parameter not valid.
CPF3CF2 E	Error(s) occurred during running of &1 API.
CPF9872 E	Program or service program &1 in library &2 ended.
CPF3C1E E	Required parameter &1 omitted.
CPF0CC1 E	Error initializing the XML parser.
CPF0CC2 E	Error found on XML input document.
CPF0CC3 E	Delete files not allowed for a packaged product.
CPF0CC7 E	Requested function not successful.
CPF0CD2 E	Application information path name not valid.

API introduced: V5R1

Generate CD-ROM Premastering Information (QLPCDINF, QlpGenCdPremasteringInfo) API

Required Parameter Group:

1	Qualified user space name	Input	Char(20)
2	Format name	Input	Char(8)
3	Distribution set map identifier	Input	Char(10)
4	CD-ROM size	Input	Binary(4)
5	CD-ROM volume identifier prefix	Input	Char(30)
6	Error code	I/O	Char(*)

Default Public Authority: *USE

Service Program: QLPCDROM

Threadsafe: No

The Generate CD-ROM Premastering Information (OPM, QLPCDINF; ILE, QlpGenCdPremasteringInfo) API allows you to do the following:

- Generate the distribution set map file

  A byte-stream file in the root directory is created and contains information about which tape files reside on each CD-ROM volume. The name of this file is \qlp*tbdnnnnnnn*, where *tbdnnnnnnn* is the name of the distribution set map identifier that is provided as an input parameter to this API. This file needs to be saved with the name QDSETMAP, using the CPYTOTAP command, to the same set of tapes prior to being sent off for mastering to CD-ROM. This file is written to each CD-ROM volume and is used by programs to ease in intelligent prompting during system installation.
- Retrieve information about the files that were saved when the job was enabled for CD-ROM premastering using the Handle CD-ROM Premastering State (QLPCDRST, QlpHandleCdState) API. (**Note:** Premastering is the set of activities done in preparation for creating a master image as a template for manufacturing a number of copies of a CD-ROM.)

  This information includes the tape file name, its corresponding CD-ROM file name (only for files needed by the OS/400 installation process, such as those saved with SAVSYS or SAVLICPGM), the CD-ROM volume it will be placed on, and the volume serial position in this set of CD-ROMs.

Authorities and Locks

API Public Authority

*EXCLUDE

QSYS Library Authority

*CHANGE

User Space Authority

*CHANGE

User Space Library Authority

*USE

User Space Lock

*EXCLRD

Required Parameter Group

Qualified user space name

INPUT; CHAR(20)

The user space that receives the generated list and the library in which it is located. The first 10 characters contain the user space name, and the second 10 characters contain the library name.

The following special values are allowed for the library name:

**CURLIB* The current library is used to locate the user space. If there is no current library, QGPL (general purpose library) is used.

**LIBL* The library list is used to locate the user space.

Format name

INPUT; CHAR(8)

The content and format of the information returned.

The possible format name follows:

PCDL0100 Premastering information

For more information, see PCDL0100 List Data Section.

Distribution set map identifier

INPUT; CHAR(10)

The distribution set map identifier that uniquely establishes this set of CD-ROMs being premastered.

This name can only include the following characters:

- Uppercase characters (A-Z)
- Numeric characters (0-9)
- Underscore character (_)

This identifier should be the same as that specified on the Handle CD-ROM Premastering State API.

CD-ROM size

INPUT; BINARY(4)

The size of the CD-ROM in megabytes that is used during mastering. This information is needed to accurately calculate where the tape files reside on CD-ROM.

CD-ROM volume identifier prefix

INPUT; CHAR(30)

The prefix of the volume identifier for the CD-ROM volumes. Each CD-ROM will be assigned a unique volume identifier starting with these 30 characters. A 2-digit numeric suffix is added to this prefix and then incremented for each CD-ROM volume in the set. For example, the volume identifiers for a 3-volume CD-ROM set when a prefix of MYCDS is used would be MYCDS01, MYCDS02, and MYCDS03.

Error code

I/O; CHAR(*)

The structure in which to return error information. For the format of the structure, see Error Code Parameter.

Format of the Generated List

The file member list consists of:

- A user area
- A generic header
- An input parameter section
- A header section
- A list data section:
  - PCDL0100 format

For details about the user area and generic header, see User Space Format for List APIs. For details about the remaining items, see the following sections. For detailed descriptions of the fields in the list returned, see Field Descriptions.

When you retrieve list entry information from a user space, you must use the entry size returned in the generic header. The size of each entry may be padded at the end. If you do not use the entry size, the result may not be valid. For examples of how to process lists, see API examples.

Input Parameter Section

Offset		Type	Field
Dec	Hex		
0	0	CHAR(10)	User space name specified
10	A	CHAR(10)	User space library name specified
20	14	CHAR(8)	Format name specified
28	1C	BINARY(4)	CD-ROM size
32	20	CHAR(10)	Distribution set map identifier
42	2A	CHAR(30)	CD-ROM volume identifier prefix

Header Section

Offset		Type	Field
Dec	Hex		
0	0	CHAR(10)	User space name used
10	A	CHAR(10)	User space library name used

PCDL0100 List Data Section

Offset		Type	Field
Dec	Hex		
0	0	CHAR(17)	Tape file label
17	11	CHAR(256)	CD-ROM file name
273	111	CHAR(32)	CD-ROM volume identifier
305	131	CHAR(3)	Reserved
308	134	BINARY(4)	CD-ROM volume serial position

Field Descriptions

CD-ROM file name. The name of the file as it will be on CD-ROM.

CD-ROM size. The size of the CD-ROM in megabytes that is used during mastering.

CD-ROM volume identifier. The CD-ROM volume identifier for the CD-ROM volume that this file will be on.

CD-ROM volume identifier prefix. The 30-character prefix of the volume identifier for the CD-ROM volumes that was specified in the call to the API.

CD-ROM volume serial position. The position in a set of CD-ROMs that this CD-ROM volume is in.

Distribution set map identifier. The distribution set map identifier that was specified in the call to the API.

Format name specified. The format name that was passed to this API on the call in the format name parameter.

Reserved. An ignored field.

Tape file label. The name of the file that was saved to tape.

User space library name specified. The name of the user space library as specified on the call to the API.

User space library name used. The actual name of the library where this user space was found.

User space name specified. The name of the user space as specified on the call to the API.

User space name used. The actual name of the user space used to store the data listed.

Error Messages

Message ID	Error Message Text
CPF24B4 E	Severe error while addressing parameter list.
CPF3C1E E	Required parameter &1 omitted.
CPF3C21 E	Format name &1 is not valid.
CPF3C36 E	Number of parameters, &1, entered for this API was not valid.
CPF3C90 E	Literal value cannot be changed.
CPF3CF1 E	Error code parameter not valid.
CPF3CF2 E	Error(s) occurred during running of &1 API.
CPF3DA4 E	CD-ROM size parameter &1 not valid.
CPF3DA5 E	File will not fit on a CD-ROM.
CPF3D9D E	Information about distribution set map not found.
CPF3D9E E	Distribution set map identifier &1 is not valid.
CPF9801 E	Object &2 in library &3 not found.
CPF9802 E	Not authorized to object &2 in &3.
CPF9803 E	Cannot allocate object &2 in library &3.
CPF9804 E	Object &2 in library &3 damaged.
CPF9807 E	One or more libraries in library list deleted.
CPF9808 E	Cannot allocate one or more libraries on library list.
CPF9810 E	Library &1 not found.
CPF9820 E	Not authorized to use library &1.
CPF9830 E	Cannot assign library &1.
CPF9872 E	Program or service program &1 in library &2 ended. Reason code &3.

API Introduced: V3R6

Generate License Key (QLZAGENK) API

Required Parameter Group:

1	Product identification	Input	Char(*)
2	Product identification format name	Input	Char(8)
3	License key input	Input	Char(*)
4	License key input format name	Input	Char(8)
5	License key output	Output	Char(*)
6	Length of license key output	Input	Binary(4)
7	License key output format name	Input	Char(8)
8	Error code	I/O	Char(*)

Default Public Authority: *USE

Threadsafe: No

The Generate License Key (QLZAGENK) API generates a license key to enable users to access a product or a feature of a product. The key is specific to the product and system information entered in this API. The resulting key is a combination of 18 letters and numbers, A-F and 0-9. To run this API, the product definition of the product you are generating the key for must exist on the system.

This command also adds the license information to the license repository. The keys are saved in the repository to keep a history of all the keys created. The repository can be queried to see what keys were generated for such things as a specific product, system, and so on. For more information about the license repository, see Retrieve License Key Information API.

Authorities and Locks

API QLZAGENK Authority

*PUBLIC(*EXCLUDE)

Required Parameter Group

Product identification

INPUT; CHAR(*)

Information that uniquely identifies the product or feature for which the license key is being generated. The structure of this information is determined by the name of the format. For more information, see LICT0100 Format.

Product identification format name

INPUT; CHAR(8)

The name of the format containing the information to identify the product.

The format name is:

LICT0100 Basic product information used as input to the API. For details, see the LICT0100 Format.

License key input

INPUT; CHAR(*)

Information that is used to generate a unique license key for the product. The structure of this information is determined by the name of the format. For more information, see LICC0100 Format.

License key input format name

INPUT; CHAR(8)

The name of the format containing the input for generating the license key.

The format name is:

LICC0100 License key information used as input to the API. For details, see LICC0100 Format.

License key output

OUTPUT; CHAR(*)

Information about the license key generated. The structure of this information is determined by the name of the format. For more information, see LICK0100 Format.

Length of license key output

INPUT; BIN(4)

The length of the license key output parameter.

License key output format name

INPUT; CHAR(8)

The name of the format containing the output for the generated license key.

The format name is:

LICK0100 License key information generated from the API. For details, see LICK0100 Format.

Error code

I/O; CHAR(*)

The structure in which to return error information. For the format of the structure, see Error Code Parameter.

LICT0100 Format

The following information uniquely describes the product or feature for which the license information is to be added. For detailed descriptions of the fields, see Field Descriptions.

Offset		Type	Field
Dec	Hex		
0	0	CHAR(7)	Product ID
7	7	CHAR(6)	License term
13	D	CHAR(4)	Feature

LICC0100 Format

The following specifies the format for the license key information used to generate the license key for the product. All fields must be specified. For detailed descriptions of the fields, see Field Descriptions.

Offset		Type	Field
Dec	Hex		
0	0	BINARY(4)	Size of license key input structure
4	4	BINARY(4)	Usage limit
8	8	CHAR(7)	Expiration date
15	F	CHAR(10)	Vendor password
25	19	CHAR(8)	Serial number
33	21	CHAR(4)	Processor group
37	25	CHAR(8)	Vendor data

LICK0100 Format

The following specifies the format for the license key generated by this product. For detailed descriptions of the fields, see Field Descriptions.

Offset		Type	Field
Dec	Hex		
0	0	BINARY(4)	Bytes returned
4	4	BINARY(4)	Bytes available
8	8	CHAR(18)	License key
26	1A	CHAR(13)	Generation date and time

Field Descriptions

Bytes available. The number of bytes of data available to be returned. All available data is returned if enough space is provided.

Bytes returned. The number of bytes of data returned.

Expiration date. The date the product license will expire. After this date, the usage limit is set to the default usage limit. No user over the usage limit is allowed to access the product or feature. A new license key must be obtained from the software provider to allow further use of the product.

CYYMMDD C is the century, YY is the year, MM is the month, and DD is the day. The date must be numeric as follows:

- Century, where 0 indicates years 19*xx* and 1 indicates years 20*xx*.
- Month may not be greater than 12.
- Day may not be greater than 31.

9999999 There is no expiration date for the product or feature.

Feature. The feature of the product to which the license key is being generated. Valid values for the feature are 5001 through 9999.

Generation date and time. The date and time that the license key was generated in the CYYMMDDHHmmSS format as follows:

C Century, where 0 indicates years 19*xx* and 1 indicates years 20*xx*.

YY Year

MM Month

DD Day

HH Hour

mm Minute

SS Second

License key. The license key generated for the product. The key will be made up of characters A-F and numbers 0-9.

License term. The extent of time the authorized usage limit for a product lasts. Each time a new license term is installed for a product, the authorized usage limit must be set by:

- Obtaining a new license key from the software provider.
- Adding the new license key to the system using the Add License Key (ADDLICKEY) command.

Possible values are:

Vx The authorized usage limit is valid for the entire version of the product or feature.

VxRy The authorized usage limit is valid for the entire release of the product or feature.

VxRyMz The authorized usage limit is valid only for the modification level of the product.

Where the x and y can be a number from 0 through 9. Z can be a number 0 through 9 or a letter A through Z.

Processor group. The processor group of the system the product or feature will be installed on. This field is left justified.

**ANY* The license key generated can be used with any processor group.

Product ID. The product ID of the product or feature to which the license information is being added.

Serial number. The serial number of the system the license key will be installed on.

Size of license key input structure. The size, in bytes, of the license key input structure contained in format LICC0100.

Usage limit. The usage limit that will be in effect when the product or feature is initially installed.

0-999999 The number of users allowed to access the product or feature.

-1 Any number of users are allowed to access the product or feature.

Vendor data. An 8 character field for vendor defined usage.

Vendor password. The software vendor's password. This password is encrypted and stored with the product. It is used in validating this Generate License Key request. It must also be the same password used when adding product license information (ADDPRDLICI command or QLZADDLI API) to this product or feature. The password must begin with an alphabetic character (A through Z, $, #, or @). This character must be followed by no more than 9 alphameric characters (A through Z, 0 through 9, $, #, @, or _).

Error Messages

Message ID	Error Message Text
CPF0CB2 E	Product identifier &1 not valid.
CPF0C4B E	Product availability object &2/&1 recovery required.
CPF0C4C E	Cannot allocate object &1 in library &2.
CPF0C4D E	Error occurred while processing object &1 in library &2.
CPF24B4 E	Severe error while addressing parameter list.
CPF3CF1 E	Error code parameter not valid.
CPF3C19 E	Error occurred with receiver variable specified.
CPF3C21 E	Format name &1 is not valid.
CPF3C24 E	Length of the receiver variable is not valid.
CPF3C90 E	Literal value cannot be changed.

CPF8191 E	Product definition &4 in &9 damaged.
CPF8193 E	Product load object &4 in &9 damaged.
CPF9E05 E	Feature &3 not valid.
CPF9E0F E	Vendor password &1 not valid.
CPF9E15 E	Error in license management function.
CPF9E40 E	Usage limit &1 not valid.
CPF9E41 E	Product &1 &2 feature &3 not found or not correctly installed.
CPF9E42 E	Vendor password not correct.
CPF9E44 E	Processor group must be specified.
CPF9E45 E	Serial number must be specified.
CPF9E46 E	License key not generated.
CPF9E54 E	License term &1 not valid.
CPF9E55 E	License repository object damaged.
CPF9E59 E	Expiration date &1 is not valid.
CPF9838 E	User profile storage limit exceeded.
CPF9872 E	Program or service program &1 in library &2 ended. Reason code &3.

API Introduced: V3R1

Top | Software Product APIs | APIs by category

Generate Program Temporary Fix Name (QPZGENNM) API

Required Parameter Group:

1	Receiver variable	Output	Char(*)
2	Length of receiver variable	Input	Binary(4)
3	PTF information	Input	Char(50)
4	Format name	Input	Char(8)
5	Error code	I/O	Char(*)

Default Public Authority: *USE

Threadsafe: No

PTF save files and cover letters are usually named Q plus the PTF ID. Sometimes a PTF for another product exists on the system and has the same ID. When this happens, another name must be selected for the PTF save file and the cover letter member name. The first 8 characters of the cover letter member name must match the PTF save file name. The Generate PTF Name (QPZGENNM) API generates a save file or cover letter member name for a PTF. Checking is done to verify that the name has not been used. A unique cover letter name is generated for each national language version (NLV).

You can use the QPZGENNM API to:

- Generate a unique name for a PTF save file in the library returned in the library name field.
- Generate a unique name for a PTF cover letter member name in the library and file returned in the library name field and file name field.

Authorities and Locks

API Public Authority

*USE

Required Parameter Group

Receiver variable

OUTPUT; CHAR(*)

The receiver variable to receive the generated name. You can specify the size of the area smaller than the format requested as long as you specify the receiver variable length parameter correctly. As a result, the API returns only the data the area can hold.

Length of receiver variable

INPUT; BINARY(4)

The length of the receiver variable. The length must be at least 8 bytes. If this value is larger than the actual receiver variable, unexpected results may occur.

PTF information

INPUT; CHAR(50)

The information about the PTF or cover letter needed to generate the name. For more information, see Format of PTF Information.

Format name

INPUT; CHAR(8)

The format of the returned information.

PTFG0100 Qualified save file name.

PTFG0200 Cover letter member, file, and library name.

For more information, see Format of Returned Information.

Error code

I/O; CHAR(*)

The structure in which to return error information. For the format of the structure, see Error Code Parameter.

Format of PTF Information

The following describes the PTF information parameter. For a detailed description of the fields in this table, see Field Descriptions.

Offset		Type	Field
Dec	Hex		
0	0	CHAR(7)	PTF ID
7	7	CHAR(7)	Product ID
14	E	CHAR(6)	Release level
20	14	CHAR(14)	Reserved
34	22	CHAR(4)	National language version
38	26	CHAR(12)	Reserved

Format of Returned Information

When you generate a name for a PTF or cover letter, the name is only unique in the library and file returned by the API. The formats below show the fields returned in the receiver variable parameter. For a detailed description of the fields in these tables, see Field Descriptions.

PTFG0100 Format

Offset		Type	Field
Dec	Hex		
0	0	BINARY(4)	Bytes returned
4	4	BINARY(4)	Bytes available
8	8	CHAR(10)	File name
18	12	CHAR(10)	Library name

PTFG0200 Format

Offset		Type	Field
Dec	Hex		
0	0	CHAR(28)	Everything from the PTFG0100 format
28	1C	CHAR(10)	Member name
38	26	CHAR(*)	Reserved

Field Descriptions

Bytes available. The number of bytes of data available to be returned. All available data is returned if enough space is provided.

Bytes returned. The number of bytes of data returned.

File name. If the format requested was PTFG0100, this is the save file name to be used to store the PTF. If the format requested was PTFG0200, this is the file where the cover letter should be stored.

Library name. The library name for the PTF or cover letter file.

Member name. The member name for the cover letter.

National language version. The national language version of the cover letter for which the member name is to be generated. This field is ignored when the format is PTFG0100. For a list of valid values, see the Globalization topic.

Product ID. The PTF is for this product.

PTF ID. The identifier of the PTF.

Release level. The version, release, and modification level of the PTF. This must be in the format V*x*R*y*M*z*. Valid values for *x* and *y* are 0 through 9, and valid values for *z* are 0 through 9 or A through Z.

Reserved. This field is ignored in the returned variable parameter. In the PTF information parameter, this field is reserved and must contain blanks.

Usage Notes

The file name is not reserved and could be in use when a succeeding function tries to use it.

A name cannot be generated for a PTF that is already in device *SERVICE known to the system. An error will be signaled if this is done. For a description of *SERVICE, see the Log Program Temporary Fix Information API.

A name cannot be generated for a cover letter member if a cover letter for the same NLV and PTF ID already exists.

Error Messages

Message ID	Error Message Text
CPF0CB3 E	Value for reserved field not valid.
CPF24B4 E	Severe error while addressing parameter list.
CPF3CF1 E	Error code parameter not valid.
CPF3C19 E	Error occurred with receiver variable specified.
CPF3C20 E	Error found by program &1.
CPF3C21 E	Format name &1 is not valid.
CPF3C24 E	Length of the receiver variable is not valid.
CPF3C90 E	Literal value cannot be changed.
CPF35BE E	Product &1 &3 not supported or installed.
CPF35D5 E	Cover letter NLV not valid.
CPF35ED E	PTF ID &1 not valid.
CPF35E9 E	PTF &1-&2 &3 already in *SERVICE.
CPF358A E	Release not valid.
CPF9872 E	Program or service program &1 in library &2 ended. Reason code &3.

API Introduced: V2R3

Handle CD-ROM Premastering State (QLPCDRST, QlpHandleCdState) API

Required Parameter Group:

1	Qualified user space name	Input	Char(20)
2	Format name	Input	Char(8)
3	Current CD-ROM premastering state	Output	Binary(4)
4	Distribution set map identifier used	Output	Char(10)
5	Requested CD-ROM premastering state	Input	Binary(4)
6	Distribution set map identifier	Input	Char(10)
7	Option	Input	Binary(4)
8	Error code	I/O	Char(*)

Default Public Authority: *EXCLUDE

Service Program: QLPCDROM

Threadsafe: No

The Handle CD-ROM Premastering State (OPM, QLPCDRST; ILE, QlpHandleCdState) API allows you to do the following:

- Enable the job for CD-ROM premastering. (**Note:** Premastering is the set of activities done in preparation for creating a master image as a template for manufacturing a number of copies of a CD-ROM.)

  When the job is in this state, all save operations (SAVOBJ, SAVLIB, SAVLICPGM, SAV, SAVDLO, SAVSYS, and so forth) result in information being stored away about the sizes of these tape files. This information is then used as input for the Generate CD-ROM Premastering Information (QLPCDINF, QlpGenCdPremasteringInfo) API when it generates the QDSETMAP byte-stream file. See the description of the Generate CD-ROM Premastering Information API for an explanation of the QDSETMAP byte-stream file. Also, the following occurs:

  - During a SAVSYS operation, special files are saved to tape that are needed on the CD-ROM when you try to perform an installation.
  - Tape-write error protection is turned off during the save operations.
- Disable the job for CD-ROM premastering

  Place the job back into a state where all save operations work as before. There are two options that can be used when disabling the job for CD-ROM premastering.

  - Leave the current information about the files saved during this job as is.

    This option gives you the ability to enable the job again later to continue save operations for a given set of CD-ROMs.

    Signing off or ending your job is equivalent to performing this option.
  - Erase the current information about the files saved during this job.

This option removes all existing information about any save operations done during this job. The next time you enable your job for CD-ROM premastering, there will be no information about this set CD-ROMs.

- Query the current CD-ROM premastering job state

Authorities and Locks

API Public Authority

*EXCLUDE

QSYS Library Authority

*CHANGE

User Space Authority

*CHANGE

User Space Library Authority

*USE

User Space Lock

*EXCLRD

Required Parameter Group

Qualified user space name

INPUT; CHAR(20)

The user space that receives the generated list and the library in which it is located. The first 10 characters contain the user space name, and the second 10 characters contain the library name.

The following special value is allowed for the user space name:

**NONE* No information about tape files that are saved when a job was enabled for CD-ROM premastering are returned.

The following special values are allowed for the library name:

**CURLIB* The current library is used to locate the user space. If there is no current library, QGPL (general purpose library) is used.

**LIBL* The library list is used to locate the user space.

This parameter must be blank when the requested CD-ROM premastering state parameter is not -1.

Format name

INPUT; CHAR(8)

The content and format of the information returned.

The possible format name follows:

TPFL0100 The list of tape files that are saved when the job was enabled for CD-ROM premastering

For more information, see TPFL0100 List Data Section. This parameter must be blank when the requested CD-ROM premastering state parameter is not -1.

Current CD-ROM premastering state

OUTPUT; BINARY(4)

The variable that receives the current CD-ROM premastering state.

The possible values follow:

0 The job is not enabled for CD-ROM premastering

1 The job is enabled for CD-ROM premastering

Distribution set map identifier used

OUTPUT; CHAR(10)

The variable that receives the distribution set map identifier for the set of tapes currently being premastered.

Requested CD-ROM premastering state

INPUT; BINARY(4)

The CD-ROM premastering state that the job should be set to.

The possible values follow:

0 Disable the job for CD-ROM premastering without destroying the information about the tape files that have been saved.

1 Enable the job for CD-ROM premastering.

2 Disable the job for CD-ROM premastering and also destroy all information about tape files that have been saved.

-1 Return the current CD-ROM premastering job state and the distribution set map identifier that are being used in the output parameters.

Distribution set map identifier

INPUT; CHAR(10)

The distribution set map identifier that uniquely establishes this set of CD-ROMs being premastered.

This name can only include the following characters:

- Uppercase characters (A-Z)

- Numeric characters (0-9)
- Underscore character (_)

This parameter must be blank when the requested CD-ROM premastering state parameter is not 1.

Option

INPUT; BINARY(4)

Whether to create new information about this particular set of CD-ROMs or to add information to a set of CD-ROMs. This parameter must be 0 when the requested CD-ROM premastering state parameter is not 1.

The possible values follow:

0 Start over with new information about CD-ROM premastering. If existing information is found about this set of CD-ROMs, the information will be erased.

1 Add information about CD-ROM premastering. If existing information is found about this set of CD-ROMs, it continues to be used and any future save operations that are done during this job are added to it.

Error code

I/O; CHAR(*)

The structure in which to return error information. For the format of the structure, see Error Code Parameter.

Format of the Generated List

The file member list consists of the following:

- A user area
- A generic header
- An input parameter section
- A header section
- A list data section:
  - TPFL0100 format

For details about the user area and generic header, see. For details about the remaining items, see the following sections. For detailed descriptions of the fields in the list returned, see Field Descriptions.

When you retrieve list entry information from a user space, you must use the entry size returned in the generic header. The size of each entry may be padded at the end. If you do not use the entry size, the result may not be valid. For examples of how to process lists, see.

Input Parameter Section

Offset		Type	Field
Dec	Hex		
0	0	CHAR(10)	User space name specified
10	A	CHAR(10)	User space library name specified
20	14	CHAR(8)	Format name specified
28	1C	BINARY(4)	Requested CD-ROM premastering state
32	20	BINARY(4)	Option
36	24	CHAR(10)	Distribution set map identifier

Header Section

Offset		Type	Field
Dec	Hex		
0	0	CHAR(10)	User space name used
10	A	CHAR(10)	User space library name used

TPFL0100 List Data Section

Offset		Type	Field
Dec	Hex		
0	0	CHAR(17)	Tape file label

Field Descriptions

Distribution set map identifier. The distribution set map identifier that was specified in the call to the API.

Format name specified. The format name that was passed to this API on the call in the format name parameter.

Option. The option that was passed to this API on the call in the option parameter.

Requested CD-ROM premastering state. The requested CD-ROM premastering state that was specified in the call to the API.

Tape file label. The name of the tape file that was saved to tape.

User space library name specified. The name of the user space library as specified on the call to the API.

User space library name used. The actual name of the library where this user space was found.

User space name specified. The name of the user space as specified on the call to the API.

User space name used. The actual name of the user space used to store the data listed.

Error Messages

Message ID	Error Message Text
CPF24B4 E	Severe error while addressing parameter list.
CPF3C1E E	Required parameter &1 omitted.
CPF3C21 E	Format name &1 is not valid.
CPF3C36 E	Number of parameters, &1, entered for this API was not valid.
CPF3C90 E	Literal value cannot be changed.
CPF3CF1 E	Error code parameter not valid.
CPF3CF2 E	Error(s) occurred during running of &1 API.
CPF3D9B E	Requested CD-ROM premastering state not valid.
CPF3D9C E	Option &1 not valid.
CPF3D9E E	Distribution set map identifier &1 is not valid.
CPF3DA2 E	&1 parameter is not blank.
CPF3DA3 E	&1 parameter is not 0.
CPF9801 E	Object &2 in library &3 not found.
CPF9802 E	Not authorized to object &2 in &3.
CPF9803 E	Cannot allocate object &2 in library &3.
CPF9804 E	Object &2 in library &3 damaged.
CPF9807 E	One or more libraries in library list deleted.
CPF9808 E	Cannot allocate one or more libraries on library list.
CPF9810 E	Library &1 not found.
CPF9820 E	Not authorized to use library &1.
CPF9830 E	Cannot assign library &1.
CPF9872 E	Program or service program &1 in library &2 ended. Reason code &3.

API Introduced: V3R6

Install Secondary Language (QLPISLNG) API

Required Parameter Group:

1	NLV language ID	Input	Char(4)
2	Device	Input	Char(10)
3	Interactive mode screen I/O indicator	Input	Char(1)
4	Error code	I/O	Char(*)

Default Public Authority: *USE

Threadsafe: No

The Install Secondary Language (QLPISLNG) API installs the secondary language that is specified in the NLV language ID parameter. It allows both an interactive mode and a batch mode of installation to install the secondary language.

Authorities and Locks

Public API authority

*EXCLUDE

User authority

*SAVSYS

Device lock

*EXCLRD

Secondary language library lock

*SHRUPD

Required Parameter Group

NLV language ID

INPUT; CHAR(4)

The national language version (NLV) language identifier of the secondary language to install.

Notes:

1. For a list of secondary languages, see the Software Installation book.
2. Both 29xx and 57xx are accepted for this parameter. The Software Installation book lists

secondary language IDs as 57xx and primary language IDs as 29xx. Therefore, to eliminate the possibility of failing the installation when the calling program passes a 57xx language ID, the API accepts both the 29xx and 57xx language IDs.

Device

INPUT; CHAR(10)

The name of the device from which to install.

Interactive mode screen I/O indicator

INPUT; CHAR(1)

An indicator as to whether screen I/O is desired in interactive mode. Valid values are:

0 Screen I/O is not desired

1 Screen I/O is desired

Note: When the API runs in batch mode, no screen I/O is provided regardless of the value of this indicator.

Error code

I/O; CHAR(*)

The structure in which to return error information. For the format of the structure, see Error Code Parameter.

Error Messages

Message ID	Error Message Text
CPD3DAC E	Device &1 is not a valid installation device.
CPF24B4 E	Severe error while addressing parameter list.
CPF3C90 E	Literal value cannot be changed.
CPF3CF1 E	Error code parameter not valid.
CPF3DAA E	NLV language ID &1 is not valid.
CPF3DAB E	Secondary language installation is not complete.
CPF3DAC E	Language ID specified is the primary language ID.
CPF3DAD E	Screen I/O parameter value is not valid.
CPF3DAE E	Not authorized to the secondary language installation process.
CPF9814 E	Device &1 not found.
CPF9872 E	Program or service program &1 in library &2 ended. Reason code &3.

API Introduced: V3R6

Top | Software Product APIs | APIs by category

List Product in a Save File (QLPLPRDS) API

Required Parameter Group:

1	Qualified user space name	Input	Char(20)
2	Format name	Input	Char(8)
3	Qualified save file name	Input	Char(20)
4	Error code	I/O	Char(*)

Default Public Authority: *USE

Threadsafe: No

The List Product in a Save File (QLPLPRDS) API generates a list for all product loads found in a save file. This list contains:

- The product ID
- The release level
- The option
- The load type
- The language ID

The products must have been saved into the save file with the SAVLICPGM command. The list is placed in the specified user space. The generated list replaces any information in the user space.

Authorities and Locks

User Space Authority

*CHANGE

Library Authority

*EXECUTE

Save File Authority

Operational

User Space Lock

*EXCL

Save File Lock

*SHRRD

Required Parameter Group

Qualified user space name

INPUT; CHAR(20)

The user space that is to receive the created list. The first 10 characters contain the user space name. The second 10 characters contain the name of the library where the user space is located.

You can use these special values for the library name:

**CURLIB* The job's current library

**LIBL* The library list

Format name

INPUT; CHAR(8)

The content and format of the information returned for each product code or language load.

The possible format names are:

PRDL0100 Product ID, Release level, Product option, Load type, Language ID.

Qualified save file name

INPUT; CHAR(20)

The name of the save file where the product was saved using SAVLICPGM. The first 10 characters contain the save file name, and the second 10 characters contain the name of the library where the file is located.

You can use these special values for the library name:

**CURLIB* The job's current library

**LIBL* The library list

Error code

I/O; CHAR(*)

The structure in which to return error information. For the format of the structure, see Error Code Parameter.

Format of the Generated Lists

The list consists of:

- A user area
- A generic header

- An input parameter section
- A header section
- A list data section:
    - PRDL0100 format

For details about the user area and generic header, see . For details about the remaining items, see the following sections. For detailed descriptions of the fields in the list returned, see Field Descriptions.

When you retrieve list entry information from a user space, you must use the entry size returned in the generic header. The size of each entry may be padded at the end. If you do not use the entry size, the result may not be valid.

Input Parameter Section

Offset		Type	Field
Dec	Hex		
0	0	CHAR(10)	User space name specified
10	A	CHAR(10)	User space library name specified
20	14	CHAR(8)	Format name
28	1C	CHAR(10)	Save file name specified
38	26	CHAR(10)	Save file library name specified

Header Section

Offset		Type	Field
Dec	Hex		
0	0	CHAR(10)	User space name used
10	A	CHAR(10)	User space library name used
20	14	CHAR(10)	Save file name used
30	1E	CHAR(10)	Save file library name used

PRDL0100 Format

Offset		Type	Field
Dec	Hex		
0	0	CHAR(7)	Product ID
7	7	CHAR(6)	Release level
13	D	CHAR(4)	Product option
17	11	CHAR(10)	Load type

27	1B	CHAR(4)	Language ID

Field Descriptions

Format name. The content and format of the information returned for each product.

The possible values are:

PRDL0100 Product ID, release level, product option, load type, language ID.

Language ID. The ID of the language found for the product in the save file. This field will be blank when the load type is *CODE.

Load type. The type of load found for the product in the save file.

The possible values are:

**CODE* indicating the load is a code load

**LNG* indicating the load is a language load

Product ID. The name of a product found in the save file whose product information is to be placed in the list.

Product option. A product option number found in the save file whose product information is to be placed in the list.

Release level. The version, release, and modification level of the product.

The format of this field is VvRrMm where:

v Product version

r Product release

m Product modification

Save file library name specified. The name of the library containing the save file whose product information is to be placed in the list.

Save file library name used. The name of the library containing the save file whose product information is placed in the list.

Save file name specified. The name of the save file specified in the call to the API.

Save file name used. The name of the save file whose product information is placed in the list.

User space library name specified. The name of the library specified in the call to the API that contains the user space.

User space library name used. The name of the library that contains the user space that is to receive the generated list.

User space name specified. The name of the space specified in the call to the API.

User space name used. The name of the user space that is to receive the generated list.

Error Messages

Message ID	Error Message Text
CPF0600 E	All CPF06xx messages could be returned. xx is from 01 to FF.
CPF24B4 E	Severe error while addressing parameter list.
CPF3CAA E	List is too large for user space &1.
CPF3CF1 E	Error code parameter not valid.
CPF3CF2 E	Error(s) occurred during running of &1 API.
CPF3C1E E	Required parameter &1 omitted.
CPF3C20 E	Error found by program &1.
CPF3C21 E	Format name &1 is not valid.
CPF3C90 E	Literal value cannot be changed.
CPF3D94 E	No product found in save file.
CPF3200 E	All CPF32xx messages could be returned. xx is from 01 to FF.
CPF8100 E	All CPF81xx messages could be returned. xx is from 01 to FF.
CPF9800 E	All CPF98xx messages could be signaled. xx is from 01 to FF.

API Introduced: V3R1

List Program Temporary Fixes (QpzListPTF) API

Required Parameter Group:

1	Qualified user space name	Input	Char(20)
2	Product information	Input	Char(50)
3	Format name	Input	Char(8)
4	Error code	I/O	Char(*)

Service Program: QPZLSTFX

Default Public Authority: *EXCLUDE

Threadsafe: No

The List Program Temporary Fixes (QpzListPTF) API returns a list of PTFs for the specified product, option, load, and release. The product must be supported or installed before the list of PTFs is returned.

Authorities and Locks

User Space Authority

*CHANGE

User Space Library Authority

*EXECUTE

User Space Lock

*EXCLRD

Required Parameter Group

Qualified user space name

INPUT; CHAR(20)

The user space that is to receive the generated list. The first 10 characters contain the user space name. The second 10 characters contain the name of the library where the user space is located. You can use these special values for the library name:

*CURLIB The job's current library.

*LIBL The library list.

Product information

INPUT; CHAR(50)

The attributes of the product for which information is being requested. For more information on this parameter, see Format of Product Information.

Format name

INPUT; CHAR(8)

The content and format of the information that is returned for each PTF. The possible format names are:

PTFL0100 List of PTFs for product, option, load, and release. For more information, see PTFL0100 Format List Section.

Error code

I/O; CHAR(*)

The structure in which to return error information. For the format of the structure, see Error Code Parameter.

Format of Product Information

For the detailed descriptions of each field, see the Field Descriptions.

Offset		Type	Field
Dec	Hex		
0	0	CHAR(7)	Product ID
7	7	CHAR(6)	Release level
13	D	CHAR(4)	Option
17	11	CHAR(10)	Load ID
27	1B	CHAR(1)	Include superseded PTFs
28	1C	CHAR(22)	Reserved

Field Descriptions

Include superseded PTFs. Whether to include superseded PTFs in the list. Valid values follow:

0 Do not include superseded PTFs in the list.

1 Include superseded PTFs in the list.

Load ID. The load ID for which PTFs are being requested. Load IDs are 10 characters in length. For example, 2924 is the load ID for an English national language version (NLV). You can use the following special value for the load ID:

*ALL The list of PTFs contains PTFs for all loads for the product specified.

Option. The option number for which PTFs are being requested. Use 0000 for the base option. Valid values are 0000 through 0099, where each character is a digit. You can use this special value for the option:

*ALL The list of PTFs contains PTFs for all options of the specified product.

Product ID. The product ID for which PTFs are being requested.

Release level. The release of the product for which PTFs are requested.

VxRyMz The release for which PTFs are being requested. The release must be in the format VxRyMz. Valid entries for x and y are any number between 0 and 9. A valid entry for z is a number between 0 and 9 or a character between A and Z.

You can use this special value for the release:

*ALL The generated list of PTFs contains PTFs for all releases of the specified product.

Reserved. This field must contain hexadecimal zeros.

Format of the Generated Lists

The user space will contain:

- A user area
- A generic header
- An input parameter section
- A header section
- A list data section:
  - PTFL0100 format

For details about the user area and generic header, see User Space Format for List APIs. For details about the remaining items, see the following sections. For detailed descriptions of the fields in the generated list returned, see Field Descriptions.

When you retrieve the list entry information from a user space, you must use the entry size returned in the generic header. The size of each entry may be padded at the end. If you do not use the entry size, the result may not be valid. For examples of how to process lists, see API examples.

Input Parameter Section

Offset		Type	Field
Dec	**Hex**		
0	0	CHAR(10)	User space name specified
10	A	CHAR(10)	User space library name specified
20	14	CHAR(50)	Product information

70	46	CHAR(8)	Format name

Header Section

Offset		Type	Field
Dec	Hex		
0	0	CHAR(10)	User space library name used
10	A	CHAR(10)	User space name used

PTFL0100 Format List Section

Offset		Type	Field
Dec	Hex		
0	0	CHAR(7)	PTF ID
7	7	CHAR(6)	Release level of the PTF
13	D	CHAR(4)	Product option of the PTF
17	11	CHAR(4)	Product load of the PTF
21	15	CHAR(1)	Loaded status
22	16	CHAR(1)	Save file status
23	17	CHAR(1)	Cover letter status
24	18	CHAR(1)	On-order status
25	19	CHAR(1)	IPL action
26	1A	CHAR(1)	Action pending
27	1B	CHAR(1)	Action required
28	1C	CHAR(1)	IPL required
29	1D	CHAR(1)	PTF is released
30	1E	CHAR(2)	Minimum level
32	20	CHAR(2)	Maximum level
34	22	CHAR(13)	Status date and time
47	2F	CHAR(7)	Superseded by PTF ID

Field Descriptions

Action pending. Indicates whether a required action has yet to be performed to make this PTF active. This field reflects the current status of any required actions.

0 No required actions are pending for this PTF.

1 A required action needs to occur for this PTF to be active. Check the Activation Instructions section of the cover letter to determine what the action is.

If the action required field is 2 and you have performed the activation instructions, then the PTF is active. However, this field will not be updated until the next IPL.

Action required. An action is required to make this PTF active when it is applied. See the cover letter to determine what action needs to be taken. The following values are valid:

0 No activation instructions are needed for this PTF.

1 This PTF was shipped with activation instructions in the cover letter. This value is returned for all PTFs that have an exit program to update the status of the PTF after the activation instructions have been performed.

2 This PTF was shipped with activation instructions in the cover letter. No exit program exists to verify the activation instructions were performed.

Cover letter status. Whether a cover letter exists for the PTF. The following values are valid.

0 The PTF has no cover letter.

1 The PTF has a cover letter.

IPL action. The action to be taken on this PTF during the next IPL. The following values are valid:

0 No action occurs at the next IPL.

1 The PTF is temporarily applied at the next IPL.

2 The PTF is temporarily removed at the next IPL.

3 The PTF is permanently applied at the next IPL.

4 The PTF is permanently removed at the next IPL.

IPL required. An IPL is required to apply this PTF. The following values are valid:

0 The PTF is delayed. The PTF must be applied during an IPL.

1 The PTF is immediate. No IPL is needed to apply the PTF.

Blank The type of the PTF is not known.

Loaded status. The current loaded status of the PTF. A PTF can have any of the following statuses:

0 The PTF has never been loaded.

1 The PTF has been loaded.

2 The PTF has been applied.

3 The PTF has been applied permanently.

4 The PTF has been permanently removed.

5 The PTF is damaged. An error occurred while applying the PTF. It needs to be reloaded and applied.

6 The PTF is superseded. » A PTF will have a status of superseded when one of the following situations occurs:

- Another PTF with a more recent correction for the problem has been loaded on the system. The PTF ID that has been loaded can be found in the superseded by PTF ID field.
- The PTF save file for another PTF with a more recent correction for the problem has been logged into *SERVICE on the system.

«

Note: These fields are returned as numbers instead of text because statuses are translatable text instead of special values. The text message that contains these values is CPX3501.

Maximum level. The indicator of the highest level of the product to which this PTF can be applied. The level can be AA to 99. This field will be blank if the product does not have a level.

Minimum level. The indicator or the lowest level of the product to which this PTF can be applied. The level can be AA to 99. This field will be blank if the product does not have a level.

On-order status. Whether the PTF has been ordered. The following values are valid:

0 The PTF has not been ordered or has already been received.

1 The PTF has been ordered.

Product load of the PTF. The load ID of the product load for the PTF.

Product option of the PTF. The option of the product to which the PTF applies.

PTF ID. The identifier of the PTF.

PTF is released. Whether the PTF save file is available for distribution to another system. This is only set to 1 when the System Manager for iSeries licensed program is on the system and the product is supported. The user needs to check the PTF save file status before using this field. Possible values follow:

0 The PTF save file cannot be distributed.

1 The PTF save file is released and can be distributed to another system.

Release level of the PTF. The release of the PTF.

Save file status. Whether a save file exists for the PTF. This field should always be checked to determine if a save file exists. The following values are valid.

0 The PTF has no save file.

1 The PTF has a save file.

Status date and time. The date and time that the PTF status was last changed. This value will be blank when the status date and time is not available. The date and time field is in the CYYMMDDHHMMSS format:

C Century, where 0 indicates years 19xx and 1 indicates years 20xx.

YY Year

MM Month

DD Day

HH Hour

MM Minute

SS Second

»**Superseded by PTF ID.** The identifier of the PTF that has replaced this PTF. This field will be blank when the PTF is not superseded or when the superseding PTF has not been loaded on the system.«

User space library name specified. The name specified for the library that contains the user space to receive the generated list.

User space library name used. The actual name of the library that is used to contain the user space that received the list.

User space name specified. The name specified for the user space that is to receive the generated list.

User space name used. The actual name of the user space that received the list.

Error Messages

Message ID	Error Message Text
CPF24B4 E	Severe error while addressing parameter list.
CPF3CF1 E	Error code parameter not valid.
CPF3CF2 E	Error(s) occurred during running of &1 API.
CPF3C1E E	Required parameter &1 omitted.
CPF3C21 E	Format name &1 is not valid.
CPF3C39 E	Value for reserved field not valid.
CPF3C4A E	Value not valid for field &1.
CPF3C90 E	Literal value cannot be changed.
CPF35BE E	Product &1 &3 not supported or installed.
CPF6601 E	No PTF activity exists for product &1.
CPF9801 E	Object &2 in library &3 not found.
CPF9802 E	Not authorized to object &2 in &3.
CPF9803 E	Cannot allocate object &2 in library &3.
CPF9820 E	Not authorized to use library &1.
CPF9838 E	User profile storage limit exceeded.

CPF9872 E	Program or service program &1 in library &2 ended. Reason code &3.

API introduced: V4R1

Top | Software Product APIs | APIs by category

List PTF Group Details(QpzListPtfGroupDetails) API

Required Parameter Group:

1	Qualified user space name	Input	CHAR(20)
2	PTF group information	Input	CHAR(*)
3	Format name	Input	CHAR(8)
4	CCSID for returned information	Input	BINARY(4)
5	Error code	I/O	CHAR(*)

Service Program Name: QPZGROUP

Default Public Authority: *USE

Threadsafe: No

The List PTF Group Details (QpzListPtfGroupDetails) API lists information for a specific PTF group on the system. The information returned is determined by the format specified.

Authorities and Locks

Work with PTF Groups (WRKPTFGRP) command

*USE

User Space Authority

*CHANGE

User Space Library Authority

*EXECUTE

User Space Lock

*EXCLRD

Lock conflicts may occur if this API is called while another PTF or PTF group operation is in progress.

Required Parameter Group

Qualified user space name

INPUT; CHAR(20)

The user space that is to receive the generated list. The first 10 characters contain the user space name.

The second 10 characters contain the name of the library where the user space is located. You can use these special values for the library name:

**CURLIB* The job's current library.

**LIBL* The library list.

PTF group information

INPUT; CHAR(*)

The attributes of the PTF group for which information is being requested. For more information on this parameter, see Format of PTF group information.

Format name

INPUT; CHAR(8)

The content and format of the information returned for this PTF group. The possible format names are:

GRPR0100 Return basic information about the PTF group. All information is returned in the user space Header Section, no list information is returned for this format. For details, see Header Section.

GRPR0200 Return basic information about the PTF group including the list of PTFs named within the PTF group. For details, see GRPR0200 Format List Section.

GRPR0300 Return basic information about the PTF group including the list of PTFs named within the PTF group and their detailed information. For details, see GRPR0300 Format List Section.

GRPR0400 Return the same information as in GRPR0300 format, but for any PTFs that are superseded by a later PTF, return detailed information about the superseded by PTF ID in place of the superseded PTF. For details, see GRPR0300 Format List Section.

GRPR0500 Return the list of related PTF groups. For details, see GRPR0500 Format List Section.

CCSID for returned information

INPUT; BINARY(4)

The coded character set ID in which to return the PTF group name, description, and related PTF group names. Valid values are 0 through 65533. If a value of 0 is specified, the names and descriptions will be returned in the CCSID of the job.

Error code

I/O; CHAR(*)

The structure in which to return error information. For the format of the structure, see Error Code Parameter.

Format of PTF group information

For the detailed descriptions of each field, see the Field Descriptions.

Offset		Type	Field
Dec	Hex		
0	0	BINARY(4)	Length of PTF group information
4	4	CHAR(60)	PTF group name
64	40	BINARY(4)	CCSID of PTF group name
68	44	CHAR(1)	Include related PTF groups

Field Descriptions

CCSID of PTF group name. The coded character set ID for the PTF group name specified in the PTF group information parameter. Valid values are 0 through 65533. If a value of 0 is specified, the name is assumed to be in the CCSID of the job.

Include related PTF groups. Whether to include information from all related PTF groups in the returned list information. The following values are valid:

0 Do not include information from related PTF groups. For formats GRPR0200, GRPR0300, and GRPR0400, only the information for the PTFs named within this PTF group is returned. For format GRPR0500, only the related PTF groups named within this PTF group are returned.

1 Include all information from all related PTF groups. For formats GRPR0200, GRPR0300, and GRPR0400, the information for all PTFs named within this PTF group and all related PTF groups is returned. For format GRPR0500, the related PTF groups in all related PTF groups are returned.

Length of PTF group information. The length of the data in the PTF group information parameter, including this field. The only valid value for this field is 69.

PTF group name. The name of the PTF group for which information is being requested. The name is limited to a maximum of 30 characters.

Format of the Generated Lists

The user space will contain:

- A user area
- A generic header
- An input parameter section
- A header section
- A list data section:
  - GRPR0200 format
  - GRPR0300 format
  - GRPR0400 format

- GRPR0500 format

For details about the user area and generic header, see User Space Format for List APIs. For details about the remaining items, see the following sections. For detailed descriptions of the fields in the generated list returned, see Field Descriptions.

When you retrieve the list entry information from a user space, you must use the entry size returned in the generic header. The size of each entry may be padded at the end. If you do not use the entry size, the result may not be valid. For examples of how to process lists, see API examples.

Input Parameter Section

Offset		Type	Field
Dec	Hex		
0	0	CHAR(10)	User space name specified
10	A	CHAR(10)	User space library name specified
20	14	CHAR(60)	PTF group name
80	50	CHAR(8)	Format name
88	58	BINARY(4)	CCSID of PTF group name
92	5C	BINARY(4)	CCSID for returned information
96	60	CHAR(1)	Include related PTF groups

Header Section

Offset		Type	Field
Dec	Hex		
0	0	CHAR(10)	User space name used
10	A	CHAR(10)	User space library name used
20	14	CHAR(60)	PTF group name
80	50	CHAR(100)	PTF group description
180	B4	BINARY(4)	PTF group level
184	B8	BINARY(4)	PTF group status

GRPR0200 Format List Section

Offset		Type	Field
Dec	Hex		
0	0	CHAR(7)	PTF ID
7	7	CHAR(7)	Product ID

14	E	CHAR(6)	Release
20	14	CHAR(4)	Product option
24	18	CHAR(4)	Product load ID
28	1C	CHAR(2)	Minimum level
30	1E	CHAR(2)	Maximum level

GRPR0300 Format List Section

Offset		Type	Field
Dec	**Hex**		
0	0	CHAR(7)	PTF ID
7	7	CHAR(7)	Product ID
14	E	CHAR(6)	Release
20	14	CHAR(4)	Product option
24	18	CHAR(4)	Product load ID
28	1C	CHAR(2)	Minimum level
30	1E	CHAR(2)	Maximum level
32	20	CHAR(1)	Loaded status
33	21	CHAR(1)	IPL action
34	22	CHAR(1)	Action pending
35	23	CHAR(1)	Action required
36	24	CHAR(1)	Cover letter status
37	25	CHAR(1)	On-order status
38	26	CHAR(1)	Save file status
39	27	CHAR(10)	Save file name
49	31	CHAR(10)	Save file library name
59	3B	CHAR(7)	Superseded by PTF ID
66	42	CHAR(7)	Latest superseding PTF ID
73	49	CHAR(1)	Product status

GRPR0500 Format List Section

Offset		Type	Field
Dec	**Hex**		
0	0	CHAR(60)	Related PTF group name
60	3C	CHAR(100)	PTF group description
160	A0	BINARY(4)	PTF group level

164	A4	BINARY(4)	PTF group status

Field Descriptions

Action pending. Indicates whether a required action has yet to be performed to make this PTF active. This field reflects the current status of any required actions. The following values are valid:

0 No required actions are pending for this PTF.

1 A required action needs to occur for this PTF to be active. Check the Activation Instructions section of the cover letter to determine what the action is.

If the action required field is 2 and you have performed the activation instructions, then the PTF is active. However, this field will not be updated until the next IPL.

Action required. An action is required to make this PTF active when it is applied. See the cover letter to determine what action needs to be taken. The following values are valid:

0 No activation instructions are needed for this PTF.

1 This PTF was shipped with activation instructions in the cover letter. This value is returned for all PTFs that have an exit program to update the status of the PTF after the activation instructions have been performed.

2 This PTF was shipped with activation instructions in the cover letter. No exit program exists to verify the activation instructions were performed.

Cover letter status. Whether a cover letter exists for the PTF. The following values are valid.

0 The PTF has no cover letter.

1 The PTF has a cover letter.

IPL action. The action to be taken on this PTF during the next IPL. The following values are valid:

0 No action occurs at the next IPL.

1 The PTF is temporarily applied at the next IPL.

2 The PTF is temporarily removed at the next IPL.

3 The PTF is permanently applied at the next IPL.

4 The PTF is permanently removed at the next IPL.

Latest superseding PTF ID. The identifier of the most recent supersede of this PTF that exists on the system. This field will be blank when the PTF does not have a superseding PTF.

Loaded status. The current loaded status of the PTF. The following values are valid:

0 The PTF has never been loaded.

1 The PTF is loaded.

2 The PTF is applied temporarily.

3 The PTF is applied permanently.

4 The PTF is permanently removed.

5 The PTF is damaged. An error occurred while applying the PTF. It needs to be reloaded and applied.

6 The PTF is superseded. A PTF will have a status of superseded when one of the following situations occurs:

- Another PTF with a more recent correction for the problem has been loaded on the system. The identifier of the PTF that has been loaded can be found in the superseded by PTF ID field.
- The PTF save file for another PTF with a more recent correction for the problem has been logged into *SERVICE on the system. The most recent PTF ID with a PTF save file that has been logged into *SERVICE can be found in the latest superseding PTF field.

Library name. The name of the library where the user space can be located. You can use these special values for the library name:

**CURLIB* The job's current library.

**LIBL* The library list.

Maximum level. The indicator of the highest level of the product on which this PTF can be installed. If the minimum and maximum levels are the same, then this PTF can only be installed on one level of the product. The level can be AA to 99. This field will be blank if the product has no level.

Minimum level. The indicator of the lowest level of the product on which this PTF can be installed. If the minimum and maximum levels are the same, then this PTF can only be installed on one level of the product. The level can be AA to 99. This field will be blank if the product has no level.

On-order status. Whether the PTF has been ordered. The following values are valid:

0 The PTF has not been ordered or has already been received.

1 The PTF has been ordered.

Product ID. The product identifier of the PTF.

Product load ID. The load ID of the product load for the PTF. A value of "CODE" indicates this PTF is for the code load of the product.

Product option. The option of the product for the PTF.

Product status. The current status of the product. The following values are valid:

0 The PTF is for a product, version, option, load identifier, and level that is not installed or supported on the system.

1 The PTF is for a product, version, option, load identifier, and level that is installed on the system. The product may or may not be supported.

2 The PTF is for a product, version, option, and load identifier, that is only supported on the system, it is not installed.

PTF group description. The text description of the PTF group.

PTF group level. The level of the PTF group. A value of 0 indicates the PTF group level cannot be determined.

PTF group name. The name of the PTF group for which information is being requested.

PTF group status. The overall status of the PTF group on this system. The PTF group status is obtained using the status of all the PTFs listed within the PTF group as well as all PTFs listed in all related PTF groups. The following values are valid:

0 Unknown. The PTF group status cannot be resolved because a related PTF group is either not found on the system or is in error.

1 Not applicable. All PTFs in the PTF group and related PTF groups are for products that are not installed or supported on this system.

2 Supported only. There are no PTFs in the PTF group or related PTF groups that are for installed products on this system. There is at least one PTF that is for a product, release, option, and load identifier that is supported on this system.

3 Not installed. There is at least one PTF that is for an installed product on this system, and not all of the PTFs or their superseding PTFs are temporarily or permanently applied.

4 Installed. All PTFs for products that are installed on this system are temporarily or permanently applied. If a PTF is superseded, a superseding PTF is either temporarily or permanently applied.

5 Error. The PTF group information is in error. Either delete the PTF group or replace the PTF group information that is currently on the system.

6 Not found. The PTF group is not found on the system. This status will only be returned for the status of a related PTF group when using format GRPR0500.

PTF ID. The identifier of the PTF in the PTF group.

Related PTF group name. The name of the related PTF group.

Release. The version, release, and modification of the product in the format V*x*R*y*M*z*. Valid values for *x* and *y* are 0 through 9, and valid values for *z* are 0 through 9 or A through Z.

Save file library name. The name of the library where the save file for the PTF is located. If no save file name has been reserved, this field will be blank.

Save file name. The name of the file where the save file for the PTF is located. You should use the save file status field to determine if a save file exists for this PTF. If no save file name have been reserved, this field will be blank.

Save file status. Whether a save file exists on the system for the PTF. The following values are valid.

0 The PTF has no save file.

1 The PTF has a save file.

Superseded by PTF ID. The identifier of the PTF that has replaced this PTF. This field will be blank when the PTF is not superseded or when the superseding PTF has not been loaded on the system.

User space library name specified. The name specified for the library that contains the user space to receive the generated list.

User space library name used. The actual name of the library that is used to contain the user space that received the list.

User space name specified. The name specified for the user space that is to receive the generated list.

User space name used. The actual name of the user space that received the list.

Error Messages

Message ID	Error Message Text
CPF0CEE E	Unable to convert data to CCSID &1.
CPF24B4 E	Severe error while addressing parameter list.
CPF36A2 E	Length of PTF group name or text too long.
CPF36A4 E	PTF group &1 not found.
CPF36A5 E	Information for PTF group &1 not complete.
CPF36AF E	PTF group function already in progress.
CPF36B1 E	Value for parameter &1 not valid.
CPF3BC7 E	CCSID &1 outside of valid range.
CPF3C1D E	Length specified in parameter &1 not valid.
CPF3C21 E	Format name &1 is not valid.
CPF3CAA E	List is too large for user space &1.
CPF3CF1 E	Error code parameter not valid.
CPF3CF2 E	Error(s) occurred during running of &1 API.
CPF9801 E	Object &2 in library &3 not found.
CPF9802 E	Not authorized to object &2 in &3.
CPF9803 E	Cannot allocate object &2 in libary &3.
CPF9820 E	Not authorized to use library &1.
CPF9838 E	User profile storage limit exceeded.
CPF9872 E	Program or service program &1 in library &2 ended. Reason code &3.

API introduced: V5R2

List PTF Groups (QpzListPtfGroups) API

Required Parameter Group:

1	Qualified user space name	Input	CHAR(20)
2	Format name	Input	CHAR(8)
3	CCSID	Input	BINARY(4)
4	Error code	I/O	CHAR(*)

Service Program Name: QPZGROUP

Default Public Authority: *USE

Threadsafe: No

The List PTF Groups (QpzListPtfGroups) API returns a list of all PTF groups that are known to the system. You can then use the QpzListPtfGroupDetails API to return detailed information for a specific PTF group.

Authorities and Locks

Work with PTF Groups (WRKPTFGRP) command

*USE

User Space Authority

*CHANGE

User Space Library Authority

*EXECUTE

User Space Lock

*EXCLRD

Lock conflicts may occur if this API is called while another PTF or PTF group operation is in progress.

Required Parameter Group

Qualified user space name

INPUT; CHAR(20)

The user space that is to receive the generated list. The first 10 characters contain the user space name. The second 10 characters contain the name of the library where the user space is located. You can use these special values for the library name:

*CURLIB The job's current library.

*LIBL The library list.

Format name

INPUT; CHAR(8)

The content and format of the information that is returned. The possible format names are:

LSTG0100 List of PTF group names, levels, descriptions, and status. For details, see LSTG0100 Format List Section.

CCSID

INPUT; BINARY(4)

The coded character set ID in which to return the PTF group names and descriptions. Valid values are 0 through 65533. If a value of 0 is specified, the names and descriptions will be returned in the CCSID of the job.

Error code

I/O; CHAR(*)

The structure in which to return error information. For the format of the structure, see Error Code Parameter.

Format of the Generated Lists

The user space will contain:

- A user area
- A generic header
- An input parameter section
- A header section
- A list data section:
  - LSTG0100 format

For details about the user area and generic header, see User Space Format for List APIs. For details about the remaining items, see the following sections. For detailed descriptions of the fields in the generated list returned, see Field Descriptions.

When you retrieve the list entry information from a user space, you must use the entry size returned in the generic header. The size of each entry may be padded at the end. If you do not use the entry size, the result may not be valid. For examples of how to process lists, see API examples.

Input Parameter Section

Offset		Type	Field
Dec	Hex		
0	0	CHAR(10)	User space name specified
10	A	CHAR(10)	User space library name specified
20	14	CHAR(8)	Format name
28	1A	BINARY(4)	CCSID

Header Section

Offset		Type	Field
Dec	Hex		
0	0	CHAR(10)	User space library name used
10	A	CHAR(10)	User space name used

LSTG0100 Format List Section

The following information is repeated for each PTF group.

Offset		Type	Field
Dec	Hex		
0	0	CHAR(60)	PTF group name
60	3C	CHAR(100)	PTF group description
160	A0	BINARY(4)	PTF group level
164	A4	BINARY(4)	PTF group status

Field Descriptions

CCSID. The coded character set ID of the returned PTF group names and descriptions that was requested.

Format name. The format of the returned information that was requested.

PTF group description. The text description of the PTF group.

PTF group level. The current level of the PTF group.

PTF group name. The name of the PTF group.

PTF group status. The overall status of the PTF group on this system. The following values are valid:

0 Unknown. The PTF group status cannot be resolved because a related PTF group is either not found on the system or is in error.

1 Not applicable. All PTFs in the PTF group and related PTF groups are for products that are not installed or supported on this system.

2 Supported only. There are no PTFs in the PTF group or related PTF groups that are for installed products on this system. There is at least one PTF that is for a product, release, option, and load identifier that is supported on this system.

3 Not installed. There is at least one PTF that is for an installed product on this system, and not all of the PTFs or their superseding PTFs are temporarily or permanently applied.

4 Installed. All PTFs for products that are installed on this system are temporarily or permanently applied. If a PTF is superseded, a superseding PTF is either temporarily or permanently applied.

5 Error. The PTF group information is in error. Either delete the PTF group or replace the PTF group information that is currently on the system.

User space library name specified. The name specified for the library that contains the user space to receive the generated list.

User space library name used. The actual name of the library that is used to contain the user space that received the list.

User space name specified. The name specified for the user space that is to receive the generated list.

User space name used. The actual name of the user space that received the list.

Error Messages

Message ID	Error Message Text
CPF0CEE E	Unable to convert data to CCSID &1.
CPF24B4 E	Severe error while addressing parameter list.
CPF36AF E	PTF group function already in progress.
CPF3BC7 E	CCSID &1 outside of valid range.
CPF3CAA E	List is too large for user space &1.
CPF3CF1 E	Error code parameter not valid.
CPF3CF2 E	Error(s) occurred during running of &1 API.
CPF3C21 E	Format name &1 is not valid.
CPF9801 E	Object &2 in library &3 not found.
CPF9802 E	Not authorized to object &2 in &3.
CPF9803 E	Cannot allocate object &2 in libary &3.
CPF9820 E	Not authorized to use library &1.
CPF9838 E	User profile storage limit exceeded.

CPF9872 E	Program or service program &1 in library &2 ended. Reason code &3.

API introduced: V5R2

Top | Software Product APIs | APIs by category

List Registered Application Information (QSZLSTRA, QszListRegAppInfo) API

Required Parameter Group:

1	Application information path name for input	Input	Char(*)
2	Application information path name for output	Input	Char(*)
3	Continuation handle input	Input	Binary(4)
4	Continuation handle output	Output	Binary(4)
5	Error code	I/O	Char(*)

Service Program Name: QSZRAIRA

Default Public Authority: *EXCLUDE

Threadsafe: No

The List Registered Application Information (QSZLSTRA, QszListRegAppInfo) API retrieves the results of a query of the OS/400 Registered Application Information Repository. Data is returned as an XML document.

Authorities and Locks

Library authority(INPUT and OUTPUT)

*EXECUTE

Authority for user space containing XML input document

*USE

Authority for user space containing XML output document

*CHANGE

User space lock (INPUT)

*EXCLRD

User space lock (OUTPUT)

*EXCL

Stream file directory authority (INPUT)

*RX

Stream file directory authority (OUTPUT)

*RWX

Authority for stream file containing XML input document

*R

Authority for stream file containing XML output document

*RW

Required Parameter Group

Application information path name input

INPUT; CHAR(*)

The path name of the object that contains the XML document with the information of the components to be listed by the API. This may be a path to a user space (*USRSPC) or a path to a stream file(*STMF). The path name should be specified in the Qlg_Path_Name_T format. If a pointer is specified in the path name format, it must be 16-byte aligned. If not, unpredictable results may occur. For more information on this structure, see Path name format. The information contained in this object should be given to the API as an XML document according to the data type definition (DTD). For a detailed description of the DTD, see Software Components DTD. For a detailed description of the information that should be contained in this object, see XML document when listing component information.

Application information path name output

OUTPUT; CHAR(*)

The path name of the object that contains the XML document listing the results of the query. This may be a path to a user space (*USRSPC) or a path to a stream file(*STMF). The path name should be specified in the Qlg_Path_Name_T format. If a pointer is specified in the path name format, it must be 16-byte aligned. If not, unpredictable results may occur. For more information on this structure, see Path name format. The information contained in this object is returned to the API caller as an XML document according to the data type definition (DTD).

Continuation handle input

INPUT; BINARY(4)

This is a special value that should be used to request the remaining entries of a previous query when a user space was provided to return the component information, but the size of the space was not enough to hold the information. This value should be taken from the Continuation Handle Output parameter returned in the previous call. This parameter must be set to 0 on the first call to this API. When you specify a continuation handle for this parameter, all other parameters must have the same values as the call to the API that generated the continuation handle. Failure to do so may result in incomplete or inaccurate information.

Continuation handle output

OUTPUT; BINARY(4)

This value indicates whether the list returned is complete or if the space provided to hold the information being returned was not enough. If this value is 0, then the list is complete. If the value is different than 0, it means that only part of the information was returned, so a second call is needed to return the remaining information. Only complete components will be returned. If a component cannot

fit in the space provided, the component will not be returned; that is, a complete XML document will be returned with as many <Component> tags as can fit in the space provided.

Error code

I/O; CHAR(*)

The structure in which to return error information. For the format of the structure, see Error Code Parameter.

XML Document when listing component information

This object may be a stream file (*STMF) or a user space (*USRSPC) object. In either case, the contents of the object must be an XML document that conforms to the specified rules.

All elements and attributes defined in the data type definition (DTD) are allowed. See below for examples of possible queries. See Software Components DTD for a detailed description of each element and attribute.

In addition, special characters can be used to increase query capabilities.

The percent sign (%) stands for an unknown string of 0 or more characters. Available options are:

%xxx	The **ends with** selection criteria, where xxx are the characters with which you want the element to end.
xxx%	The **starts with** selection criteria, where xxx are the characters with which you want the element to start.
%xxx%	The **contains** selection criteria, where xxx are the characters you want to be contained in your element.

An underline character (_) stands for any single character.

If you do not specify any of these characters, it indicates that you want an exact match.

The rules to follow in your XML document to do the query are:

1. When product, version, component name, instance feature or vendor are not present or are empty, it indicates that you want any match on those elements. For example, if you do not specify Version, ComponentName, Instance and Vendor attributes or they are empty, then all applications that match the product specified will be returned. In the example below, all entries matching a 5769SS1 product will be returned (no extended data will be returned since the ExtendedData element was not present). The ComponentName attribute was specified as empty since it is required by the DTD.

   ```
   <?xml ComponentVersion="1.0" encoding='ucs-2'?>
   <!DOCTYPE RegAppInfoRepository SYSTEM
   "/IBM/XML/DTD/RegAppInfoRepository.dtd">
   <RegAppInfoRepository DTDVersion="1.0">
     <Component ProductName="5769SS1" ComponentName=""> </Component>
   </RegAppInfoRepository>
   ```

2. When you specify the PackagedProduct attribute with a value of 1, it indicates that you want only information about OS/400 packaged products. For example, if you specify the starts with selection criteria as shown in the following XML document, then all OS/400 packaged product components with a ComponentName starting with 5 will be returned. Note that the Product attribute was defined because

it is required by the DTD.

```
<?xml ComponentVersion="1.0" encoding='ucs-2'?>
<!DOCTYPE RegAppInfoRepository SYSTEM
"/IBM/XML/DTD/RegAppInfoRepository.dtd">
<RegAppInfoRepository DTDVersion="1.0">
  <Component ProductName="" ComponentName="5%" PackagedProduct="1">
  </Component>
</RegAppInfoRepository>
```

3. When you specify one of the allowed attributes or elements contained in the <ExtendedData> element and one of the attributes or elements is empty, it indicates that you want that data retrieved. For example, if you want to know if a component is supported, you can specify the ExtendedData element as follows:

```
<?xml ComponentVersion="1.0" encoding='ucs-2'?>
<!DOCTYPE RegAppInfoRepository SYSTEM
"/IBM/XML/DTD/RegAppInfoRepository.dtd">
<RegAppInfoRepository DTDVersion="1.0">
  <Component ProductName="5769XD1" ComponentVersion="V5R1M0"
    ComponentName="5050" FeatureName="*BASE" PackagedProduct="1">
    <ExtendedData Supported="">
    </ExtendedData>
  </Component>
</RegAppInfoRepository>
```

4. When you specify one of the allowed attributes or elements contained in the <ExtenedeData> element and it contains a value, it indicates that you want a component or components that have the attribute or element specified returned and which value matches (depending on the selection criteria). For example, if you want to know all the components that were installed by ISJE:

```
<?xml ComponentVersion="1.0" encoding='ucs-2'?>
<!DOCTYPE RegAppInfoRepository SYSTEM
"/IBM/XML/DTD/RegAppInfoRepository.dtd">
<RegAppInfoRepository DTDVersion="1.0">
  <Component ProductName="" ComponentName="">
    <ExtendedData InstallerType="ISJE"> </ExtendedData>
  </Component>
</RegAppInfoRepository>
```

5. When you specify the ExtendedData element with no attributes or elements inside, it indicates that you want returned all available information for that component.

```
<?xml ComponentVersion="1.0" encoding='ucs-2'?>
<!DOCTYPE RegAppInfoRepository SYSTEM
"/IBM/XML/DTD/RegAppInfoRepository.dtd">
<RegAppInfoRepository DTDVersion="1.0">
  <Component ProductName="5769XD1" ComponentVersion="V5R1M0"
             ComponentName="5050" FeatureName="*BASE" >
       <ExtendedData/>
  </Component>
</RegAppInfoRepository>
```

Additional examples illustrating the way to define an XML document to call this API follow:

List all information for all OS/400 packaged products

```
<?xml ComponentVersion="1.0" encoding='ucs-2'?>
<!DOCTYPE RegAppInfoRepository SYSTEM
"/IBM/XML/DTD/RegAppInfoRepository.dtd">
<RegAppInfoRepository DTDVersion="1.0">
  <Component ProductName="" ComponentName="" PackagedProduct="1">
     <ExtendedData>
     </ExtendedData>
  </Component>
</RegAppInfoRepository>
```

List all OS/400 packaged products with a product name ending in "XD1".

```
<?xml ComponentVersion="1.0" encoding='ucs-2'?>
<!DOCTYPE RegAppInfoRepository SYSTEM
"/IBM/XML/DTD/RegAppInfoRepository.dtd">
<RegAppInfoRepository DTDVersion="1.0">
  <Component ProductName="%XD1" ComponentName="" PackagedProduct="1"/>
</RegAppInfoRepository>
```

List all OS/400 packaged products having 57 in the product name as the first characters and 'XD1' as the last characters.

```
<?xml ComponentVersion="1.0" encoding='ucs-2'?>
<!DOCTYPE RegAppInfoRepository SYSTEM
"/IBM/XML/DTD/RegAppInfoRepository.dtd">
<RegAppInfoRepository DTDVersion="1.0">
  <Component ProductName="57__XD1" ComponentName="" PackagedProduct="1"/>
</RegAppInfoRepository>
```

Notice in this example the use of the underline character, in this case we know the product name for OS/400 packaged products is 7 characters long. Underline character should not be used if you do not know the number of characters in between.

List all available information for components containing the string tool in the component name.

```
<?xml ComponentVersion="1.0" encoding='ucs-2'?>
<!DOCTYPE RegAppInfoRepository SYSTEM
"/IBM/XML/DTD/RegAppInfoRepository.dtd">
<RegAppInfoRepository DTDVersion="1.0">
  <Component ProductName="" ComponentName="%tool%">
     <ExtendedData></ExtendedData>
  </Component>
```

```
</RegAppInfoRepository>
```

List all supported OS/400 packaged products.

Having the following XML document, we can get the list of all supported OS/400 packaged products.

```
<?xml ComponentVersion="1.0" encoding='ucs-2'?>
<!DOCTYPE RegAppInfoRepository SYSTEM
"/IBM/XML/DTD/RegAppInfoRepository.dtd">
<RegAppInfoRepository DTDVersion="1.0">
  <Component ProductName="" ComponentName="" PackagedProduct="1">
    <ExtendedData Supported="1">
    </ExtendedData>
  </Component>
</RegAppInfoRepository>
```

Error Messages

Message ID	Error Message Text
CPF24B4 E	Severe error addressing parameter list.
CPF3CF1 E	Error code parameter not valid.
CPF3CF2 E	Error(s) occurred during running of &1 API.
CPF9872 E	Program or service program &1 in library &2 ended.
CPF3C1E E	Required parameter &1 omitted.
CPF0CC1 E	Error initializing the XML parser.
CPF0CC2 E	Error found on XML input document.
CPF0CC7 E	Requested function not successful.
CPF0CD2 E	Application information path name not valid.
CPF0CDF E	Continuation handle not valid.

API introduced: V5R1

Log Program Temporary Fix Information (QPZLOGFX) API

Required Parameter Group:

1	PTF Information	Input	Char(50)
2	Request type	Input	Char(10)
3	Qualified file name	Input	Char(20)
4	Member name	Input	Char(10)
5	Error code	I/O	Char(*)

Default Public Authority: *USE

Threadsafe: No

The Log PTF Information (QPZLOGFX) API allows you to specify the device from which PTFs are loaded as *SERVICE. You can use the QPZLOGFX API to indicate that a PTF or cover letter should be put into device *SERVICE.

PTFs are put into *SERVICE when they are received on the system using the Copy PTF Save File (CPYPTFSAVF) or Send PTF Order (SNDPTFORD) command. These commands put information about the PTF into the PTF database files so that the PTF can be displayed with the Display PTF (DSPPTF) command. The PTF is then loaded from device *SERVICE using the Load PTF (LODPTF) command. PTFs received on the system by any other method are not in *SERVICE.

The PTF must exist in a save file with a valid name in the designated PTF library before this API is called. If the PTF is put in a save file with a name that is being used by another PTF, an error (CPF35BD) occurs. The save file must be deleted or renamed. Checking is done to ensure that the correct library is being used.

The cover letter must exist in the designated PTF cover letter file with a valid member name before this API is called. If the member name is being used by another PTF, an error (CPF35FE) will occur; you must delete the member or rename it.

The product the PTF is for must be installed or supported.

The save file or member name must be Q plus the PTF ID. If that name is being used by another PTF that is in *SERVICE, you must select another name.

Authorities and Locks

API Public Authority

*EXCLUDE

Required Parameter Group

PTF information

INPUT; CHAR(50)

The information needed to put the PTF into device *SERVICE. If the information specified here does not match the information in the member or save file specified on the qualified file name parameter, an error occurs (CPF35E6). A check is done to verify that the PTF ID, product ID, and release specified here match the PTF ID, product ID, and release of the cover letter or PTF. For more information about this parameter see PTF Information Format.

Request type

INPUT; CHAR(10)

The type of log operation to be done.

The possible values are:

**LOGPTF* The PTF is to be put in device *SERVICE. Product information from the save file is checked against the product information passed into this API. Therefore, the PTF must exist in the save file specified on the qualified file name parameter.

**LOGCVR* A cover letter is to be put in device *SERVICE. The cover letter must exist in the library, file, and member specified in the qualified file name and member name parameters. Information from the member is checked against the product information passed into the API.

Qualified file name

INPUT; CHAR(20)

The file where the cover letter or PTF is located. If the request type parameter is *LOGPTF, then this is the save file name and library name where the PTF is located. If the request type is *LOGCVR, then this is the file and library that contain the cover letter. The first 10 characters are the file name and the second 10 characters are the library name.

Member name

INPUT; CHAR(10)

The member that contains the cover letter. This parameter is ignored if the request type parameter is *LOGPTF.

Error code

I/O; CHAR(*)

The structure in which to return error information. For the format of the structure, see Error Code Parameter.

PTF Information Format

For a description of these fields, see Field Descriptions.

Offset		Type	Field
Dec	Hex		
0	0	CHAR(7)	PTF ID
7	7	CHAR(7)	Product ID
14	E	CHAR(6)	Release level
20	14	CHAR(30)	Reserved

Field Descriptions

Product ID. The product that the PTF or the cover letter is for.

PTF ID. The identifier of the PTF.

Release level. The version, release, and modification level of the PTF. The release must be in the format V*x*R*y*M*z*. Valid values for *x* and *y* are 0 through 9, and valid values for *z* are 0 through 9 or A through Z.

Reserved. This field must be blank.

Error Messages

Message ID	Error Message Text
CPF0CB3 E	Value for reserved field not valid.
CPF24B4 E	Severe error while addressing parameter list.
CPF3CF1 E	Error code parameter not valid.
CPF3C20 E	Error found by program &1.
CPF3C90 E	Literal value cannot be changed.
CPF35BD E	File name &4 not valid for PTF &1-&2 &3.
CPF35BE E	Product &1 &3 not supported or installed.
CPF35BF E	File name or library not valid.
CPF35DE E	Request type &1 not valid.
CPF35E5 E	PTF save file &1 in library &2 does not exist.
CPF35E6 E	File or member not processed.
CPF35FE E	Member name &7 not valid for PTF &1-&2 &3.

CPF358A E	Release not valid.
CPF3903 E	Cover letter not found.
CPF3905 E	Member not valid cover letter.
CPF9872 E	Program or service program &1 in library &2 ended. Reason code &3.

API Introduced: V2R3

Top | Software Product APIs | APIs by category

Package Product Option (QSZPKGPO) API

Required Parameter Group:

1	Product option information	Input	Char(35)
2	Repackage	Input	Char(4)
3	Allow object change	Input	Char(5)
4	Error code	I/O	Char(*)

Default Public Authority: *USE

Threadsafe: No

The Package Product Option (QSZPKGPO) API packages one or more product loads for a specified product option. This enables the loads to be saved using the Save Licensed Program (SAVLICPGM) command. Object lists are added to the product load (*PRDLOD) object when the product load is packaged. In order to package a product load, all development libraries for the load must exist and all development folders for the load must exist. Also, all objects to be packaged must exist and all objects except folders and documents must have the following correct information in the object description:

- Product ID.
- Release level.
- Product option.
- Load ID.

The object description can be changed using the Change Object Description (QLICOBJD) API.

To package a product load that has folders, the user must be enrolled in the system distribution directory and have *ALL authority to the folders.

Other than these restrictions, a user who has authority to this API can package any product load created in either of the following ways. This is regardless of whether the user has authority to the objects for the product load.

- The Create Product Load (QSZCRTPL) API
- System Manager for OS/400 licensed program

Authorities and Locks

Authority to Folders

*ALL

Authority to Objects Packaged

None

Document Lock

*SHRRD. A lock of *SHRNUP is obtained on each document's attributes.

Table of Contents for Subordinate Folders Lock

*SHRNUP

Folder Lock

*SHRUPD

Object Lock

*EXCLRD. If *NO is specified for the allow object change parameter, each object packaged is locked *EXCLRD.

Product Availability Lock

*SHRRD

Product Definition Lock

*EXCLRD

Product Load Lock

*EXCLRD

Object Lock

Each object in each of the product load's libraries is locked as follows:

- Message queues are locked *EXCLRD.
- All other object types are locked *SHRRD.

Required Parameter Group

Product option information

INPUT;CHAR(35)

The product loads to be packaged. For more information, see Product Option Information Format.

Repackage

INPUT; CHAR(4)

Whether or not to package product loads that were already packaged.

**NO* Packages product loads that have not already been packaged.

**YES* Packages all specified product loads, regardless of whether they are already packaged.

Allow object change

INPUT; CHAR(5)

Whether to prevent changes to the product information in the object description of each object added to

an object list. You can change the allow change by program attribute by using the allow change by program field of the Change Object Description (QLICOBJD) API. See the allow change by program field in the QLICOBJD API for more information.

**SAME* Does not change the allow change by program attribute of the objects being packaged.

**NO* Changes the allow change by program attribute of the objects being packaged so that the product information in the object description cannot be changed by the Change Object Description (QLICOBJD) API.

Error code

I/O; CHAR(*)

The structure in which to return error information. For the format of the structure, see Error Code Parameter.

Product Option Information Format

The following describes the product option information parameter. For a detailed description of the fields in this table, see Field Descriptions.

Offset		Type	Field
Dec	Hex		
0	0	CHAR(4)	Product option
4	4	CHAR(7)	Product ID
11	B	CHAR(6)	Release level
17	11	CHAR(8)	Load ID
25	19	CHAR(10)	Reserved

Field Descriptions

Load ID. Specifies which loads to package. Load IDs are 4 characters in length; for example, 2924 is the load ID for an English National Language Version (NLV).

The following special values are valid:

**CODEDFT* The default code load ID, 5001, is used.

**ALL* For the specified product ID, release level, and product option, all product loads for which a product load object exists are packaged. If only a code load exists, the code load is packaged.

Product ID. The product ID of the product option to be packaged.

Product option. The product option to be packaged. Use 0000 for the base option. Valid values are 0000 through 0099, where each character is a digit.

Release level. The version, release, and modification level of the product option to be packaged. The release

level must be a valid special value or it must be in the form of VxRyMz. Valid entries for x and y are 0 through 9. Valid entries for z is 0 through 9 or A through Z.

The following special value is valid:

**ONLY* The release level is determined by searching the system for a product definition (*PRDDFN) for the specified product ID. The release level is taken from the product definition. This value is not valid if there are product definitions for two or more release levels of the product on the system.

Reserved. This field must contain blanks; otherwise, an error occurs.

Error Messages

Message ID	Error Message Text
CPF0CB2 E	Product identifier &1 not valid.
CPF0CEA E	Loads not packaged for product &1 release &2 option &3.
CPF0CEB E	Product loads not packaged.
CPF0CEC E	Code load has no folders.
CPF0CED E	Reserved fields in parameter 1 are not blank.
CPF0CE0 E	Message file &5 in library &6 not found.
CPF0CE1 E	&1 not valid for repackage parameter.
CPF0CE3 E	&9 product loads packaged, &10 product loads not packaged. See job log.
CPF0CE4 E	Secondary language product load not packaged.
CPF0CE5 E	&1 not valid for allow change parameter.
CPF0CE6 E	Public authority for library &6 not valid for packaging.
CPF0CE7 E	Product &1 release &2 not packaged.
CPF0CE8 E	Primary folder list not correct.
CPF0CE9 E	Development folder list not correct.
CPF0CFA E	Load &4 for option &3 not in product definition.
CPF0CFB E	Language load not packaged.
CPF0CFC E	Product definition not found.
CPF0CFD E	Code load not found for product &1 release &2 option &3.
CPF0CFE E	Object &5 type *&7 in &6 not in development library &8.
CPF0CFF E	Multiple releases available.
CPF0CF1 E	Object &5 in &6 type *&7 associated with PTF &8.

CPF0CF4 E Object description not correct.

CPF0CF5 E Exit program &5 in library &6 not found.

CPF0CF6 E Packaging operation failed for &5 in &6 type *&7.

CPF0CF7 E Product &1 release &2 option &3 load &4 already packaged.

CPF0C4B E Product availability object &2/&1 recovery required.

CPF0C4C E Cannot allocate object &1 in library &2.

CPF0C4D E Error occurred while processing object &1 in library &2.

CPF0C8A E Product option &1 not valid.

CPF0C84 E Load identifier &4 not valid.

CPF2150 E Object information function failed.

CPF2151 E Operation failed for &2 in &1 type *&3.

CPF2225 E Not able to allocate internal system object.

CPF2352 E Program &1 in &2 received wrong parameters.

CPF24B4 E Severe error while addressing parameter list.

CPF2451 E Message queue &1 is allocated to another job.

CPF3C90 E Literal value cannot be changed.

CPF3CF1 E Error code parameter not valid.

CPF358A E Release not valid.

CPF7304 E File &1 in &2 not changed.

CPF8A06 E Document &2 or folder &3 partially created in folder &1.

CPF8A75 E Not authorized to access folder &1.

CPF8A77 E Folder &1 not found.

CPF8A78 E Folder &1 in use.

CPF8A79 E Folder &1 is logically damaged.

CPF8100 E All CPF81xx messages could be returned. xx is from 01 to FF.

CPF9012 E Start of document interchange session not successful for &1.

CPF9801 E Object &2 in library &3 not found.

CPF9803 E Cannot allocate object &2 in library &3.

CPF9804 E Object &2 in library &3 damaged.

CPF9806 E Cannot perform function for object &2 in library &3.

CPF9809 E	Library &1 cannot be accessed.
CPF9810 E	Library &1 not found.
CPF9811 E	Program &1 in library &2 not found.
CPF9812 E	File &1 in library &2 not found.
CPF9814 E	Device &1 not found.
CPF9830 E	Cannot assign library &1.
CPF9831 E	Cannot assign device &1.
CPF9872 E	Program or service program &1 in library &2 ended. Reason code &3.

API Introduced: V2R3

Top | Software Product APIs | APIs by category

Release License (QLZARLS) API

Required Parameter Group:

1	Product identification	Input	Char(*)
2	Product identification format name	Input	Char(8)
3	License user	Input	Char(*)
4	License user format name	Input	Char(8)
5	Error code	I/O	Char(*)

Default Public Authority: *USE

Threadsafe: No

The Release License (QLZARLS) API releases the use of a product from the assigned license user. This API works the opposite of the Request License (QLZAREQ) API. Whatever was previously requested is now released and the usage count is decremented. When using this API, specify the same parameters as on the Request License API.

Authorities and Locks

Public API Authority

*USE

Required Parameter Group

Product identification

INPUT; CHAR(*)

Information that uniquely identifies the product or feature whose licensed use is to be released. The structure of this information is determined by the name of the format.

Product identification format name

INPUT; CHAR(8)

The name of the format that describes the product identification.

The only format name supported is:

LICP0100 See LICP0100 Format.

License user

INPUT; CHAR(*)

The name to which use of the license product is assigned and tracked.

License user format name

INPUT; CHAR(8)

The name of the format that describes the license user.

The formats supported are:

LICL0100 See LICL0100 Format.

LICL0200 See LICL0200 Format.

Error code

I/O; CHAR(*)

The structure in which to return error information. For the format of the structure, see Error Code Parameter.

LICP0100 Format

To release the licensed use of a product or feature, you must specify the same information that was specified on the Request License (QLZAREQ) API. The following table identifies the product or feature whose licensed use is to be released. For a detailed description of the fields in this table, see Field Descriptions.

Offset		Type	Field
Dec	Hex		
0	0	CHAR(7)	Product ID
7	7	CHAR(6)	Release level
13	D	CHAR(4)	Feature

LICL0100 Format

To release the licensed use of a product or feature, you must specify the same information that was specified on the Request License (QLZAREQ) API. The following table identifies the license user whose licensed use is to be released. For a detailed description of the field in this table, see Field Descriptions.

Offset		Type	Field
Dec	Hex		
0	0	CHAR(10)	License user

LICL0200 Format

To release the licensed use of a product or feature, you must specify the same information that was specified on the Request License (QLZAREQ) API. The following table identifies the license user to be released and the license user handle. For a detailed description of the fields in this table, see Field Descriptions.

Offset		Type	Field
Dec	**Hex**		
0	0	BINARY(4)	Offset to license user
4	4	BINARY(4)	Length of license user
8	8	CHAR(8)	License user handle
16	10	BINARY(4)	Offset to additional license user information
20	14	BINARY(4)	Length of additional license user information
24	18	BINARY(4)	Reserved
Note: Use the offset to license user field to determine the offset.			
		CHAR(*)	License user
Note: Use the offset to additional license user information field to determine the offset.			
		BINARY(4)	Number of uses held

Field Descriptions

Feature.

The feature of the product. Valid values for the feature are 5001 through 9999.

Length of additional license user information. The length of the additional license user information. Set this field to 0 if there is no additional license user information.

Length of license user. The length of the license user field. This may be a value of 1 through 80. This value may only change within the same license term if there are no current users at the time the change request is made. To conserve space, use the smallest value necessary for this length.

License user. The license user being released. For registered use, specify the nonblank name you want assigned as the license user. For concurrent use, specify *JOB. For processor use, specify *PROCESSOR.

License user handle. The value passed in by the vendor at the time of the Request License (QLZAREQ) API call. When the vendor wants to release the user, this handle must be specified on the Release License (QLZARLS) API call. This value many be any 8-character field, including blanks. If the value specified on the release call does not match the value initially specified on the request call, the license user is not released. This field is never displayed or output through any license management interfaces and is only known to the vendor application.

Number of uses held. The number of uses held by the license user that are to be released. If this field is not specified, the number of uses is assumed to be 1.

The valid values follow:

1-999999 The number of uses to be released. This must be the same number of uses that were requested for the license user on the Request License (QLZAREQ) API.

-1 All of the uses held by the license user are to be released. This value is to be used if the calling program does not know the number of uses that the license user currently holds.

Offset to additional license user information. The offset from the beginning of the license user information to the additional information for the first license user. Specify 0 for this field if there is no additional information for the license user.

Offset to license user. The offset from the beginning of the license user information to the first license user.

Product ID. The product ID of the product or feature whose licensed use is to be released.

Release level. The version, release, and modification level of the product or feature. The release level must be a valid special value, or the release level must be in the format V*x*R*y*M*z*. Valid values for *x* and *y* are 0 through 9. Valid values for *z* are 0 through 9 and A through Z.

The valid special value is:

**ONLY* The release level is determined by searching the system for a given product. The release level is taken from the product definition. This value is not valid if more than one product definition exists for the same product ID.

Reserved. Reserved for future use. This field must be set to hexadecimal zeros.

Error Messages

Message ID	Error Message Text
CPF24B4 E	Severe error while addressing parameter list.
CPF3CF1 E	Error code parameter not valid.
CPF3C21 E	Format name &1 is not valid.
CPF3C90 E	Literal value cannot be changed.
CPF9E1C E	License user parameter not valid.
CPF9E1E E	Length of license user is not correct.
CPF9E1F E	Attempt made to change length of the license user, resulting code &1.
CPF9E11 E	License information not retrieved.
CPF9E12 E	License information not available.
CPF9E13 E	More than one release found.
CPF9E14 E	License user handle not correct.
CPF9E15 E	Error in license management function.
CPF9E7B E	Cannot release different number of uses than requested.

CPF9E91 E	License user &1 is not valid.
CPF9872 E	Program or service program &1 in library &2 ended. Reason code &3.
CPI9E75 I	Grace period will expire in &5 days on &4.
CPI9E76 I	Expiration date will be reached in &5 days on &4.
CPI9E77 I	License key will not be valid in &8 days on &9.

API Introduced: V2R3

Top | Software Product APIs | APIs by category

Request License (QLZAREQ) API

Required Parameter Group:

1	Product identification	Input	Char(*)
2	Product identification format name	Input	Char(8)
3	License user	Input	Char(*)
4	License user format name	Input	Char(8)
5	Error code	I/O	Char(*)

Default Public Authority: *USE

Threadsafe: No

The Request License (QLZAREQ) API requests the use of a product that has been packaged for licensed use. Multiple uses of a product may be requested with a single call to this API. The request causes the usage count to be compared with the usage limit. The uses are assigned to the name that is specified in the license user parameter. The uses remain assigned to the license user until they are released (see

Note: It is suggested that before a Request License (QLZAREQ) is done, the application retrieves the product information handle using the Retrieve License Information API (QLZARTV). The application should then compare this handle with the handle passed back at Add Product License Information time. It should compare the two values before the Request is done. If the handles do not match, the product has been tampered with. This process helps to ensure asset protection.

Authorities and Locks

Public API Authority

*EXCLUDE

Required Parameter Group

Product identification

INPUT; CHAR(*)

Information that uniquely identifies the product or feature whose licensed use is requested. The structure of this information is determined by the name of the format.

Product identification format name

INPUT; CHAR(8)

The name of the format that describes the product identification.

The only format name supported is:

LICP0100 This format identifies the product ID, release level, and feature whose use is requested. See LICP0100 Format.

License user

INPUT; CHAR(*)

The name to which use of the license product is assigned and tracked.

License user format name

INPUT; CHAR(8)

The name of the format that describes the license user information.

The formats supported are:

LICL0100 This format identifies the license user to be requested. See LICL0100 Format.

LICL0200 This format identifies the license user being requested and the multiple uses (number of uses) for each license user. See LICL0200 Format.

Error code

I/O; CHAR(*)

The structure in which to return error information. For the format of the structure, see Error Code Parameter.

LICP0100 Format

For a detailed description of the fields in this table, see Field Descriptions.

Offset		Type	Field
Dec	Hex		
0	0	CHAR(7)	Product ID
7	7	CHAR(6)	Release
13	D	CHAR(4)	Feature

LICL0100 Format

For a detailed description of the field in this table, see Field Descriptions.

Offset		Type	Field
Dec	Hex		
0	0	CHAR(10)	License user

LICL0200 Format

For a detailed description of the fields in this table, see Field Descriptions.

Offset		Type	Field
Dec	Hex		
0	0	BINARY(4)	Offset to license user
4	4	BINARY(4)	Length of license user
8	8	CHAR(8)	License user handle
16	10	BINARY(4)	Offset to additional license user information
20	14	BINARY(4)	Length of additional license user information
24	18	BINARY(4)	Reserved
Note: Use the offset to license user field to determine the offset.			
		CHAR(*)	License user
Note: Use the offset to additional license user information field to determine the offset.			
		BINARY(4)	Number of uses requested

Field Descriptions

Feature. The feature of the product. Valid values for the feature are 5001 through 9999.

Length of additional license user information. The length of the additional license user information. Set this field to 0 if there is no additional license user information.

License user. The license user being requested. For registered use, specify the nonblank name you want assigned as the license user. For concurrent use, specify *JOB. For processor use, specify *PROCESSOR.

License user handle. This value is passed in by the vendor at the time of the Request License (QLZAREQ) API call. When the vendor wants to release the user, this handle must be specified on the Release License (QLZARLS) API call. This value many be any eight character field, including blanks. If the value specified on the release call does not match the value initially specified on the request call, the license user is not released. This field is never displayed or output through any license management interfaces and is only known to the vendor application.

Length of license user. The length of the license user field. This may be a value 1 through 80. This value may only change within the same license term if there are no current users at the time the change request is made. To conserve space, use the smallest value necessary for this length.

Number of uses requested. The number of license uses being requested for this license user. This value must be between 1 and 999 999. If this field is not specified, the number of uses requested is 1.

Offset to additional license user information. The offset from the beginning of the license user information to the additional information for the first license user. Specify 0 for this field if there is no additional information for the license user.

Offset to license user. The offset from the beginning of the license user information to the first license user.

Product ID. The product ID of the product or feature whose licensed use is requested.

Release level. The version, release, and modification level of the product or feature. The release level must be a valid special value, or the release level must be in the format V*x*R*y*M*z*. Valid values for *x* and *y* are 0 through 9. Valid values for *z* are 0 through 9 and A through Z.

The valid special value is:

**ONLY* The release level is determined by searching the system for a given product. The release level is taken from the product definition. This value is not valid if more than one product definition exists for the same product ID.

Reserved. Reserved for future use. This field must be set to hexadecimal zeros.

Error Messages

Message ID	Error Message Text
CPF24B4 E	Severe error while addressing parameter list.
CPF3CF1 E	Error code parameter not valid.
CPF3C21 E	Format name &1 is not valid.
CPF3C90 E	Literal value cannot be changed.
CPF9E1C E	License user parameter not valid.
CPF9E1E E	Length of license user is not correct.
CPF9E1F E	Attempt made to change length of the license user, resulting code &1.
CPF9E11 E	License information not retrieved.
CPF9E12 E	License information not available.
CPF9E13 E	More than one release found.
CPF9E15 E	Error in license management function.
CPF9E17 E	Usage limit exceeded for product &1. User added.
CPF9E18 E	Attempt made to exceed usage limit for product &1. User not added.
CPF9E70 E	Grace period expired. Requesting user already added.
CPF9E71 E	Grace period expired. Requesting user not added.
CPF9E72 E	Usage limit of &4 exceeded. Grace period will expire in &6 days on &5.
CPF9E73 E	Expiration date &4 was reached.
CPF9E78 E	The license key for product &1, license term &2, feature &3 is no longer valid.
CPF9E79 E	Cannot specify different number of uses than already requested.

CPF9E91 E	License user &1 is not valid.
CPF9872 E	Program or service program &1 in library &2 ended. Reason code &3.
CPI9E75 I	Grace period will expire in &5 days on &4.
CPI9E76 I	Expiration date will be reached in &5 days on &4.
CPI9E77 I	License key will not be valid in &8 days on &9.

API Introduced: V2R3

Top | Software Product APIs | APIs by category

Request Order Assistance (QMARQSOA) API

Required Parameter Group:

1	Output order ID	Output	Char(31)
2	Input order ID	Input	Char(31)
3	Send or receive option	Input	Binary(4)
4	Contact information	Input	Char(*)
5	Configuration file	Input	Char(10)
6	Network address	Input	Char(*)
7	Request information	Input	Char(*)
8	System information	Input	Char(*)
9	Error code	I/O	Char(*)

Default Public Authority: *USE

Threadsafe: No

The Request Order Assistance (QMARQSOA) API sends a request for order assistance to a service provider using an ECS connection. This API also creates a corresponding entry in the user's order log. Optionally, a system configuration file can be sent along with the order assistance request. This file will help the service provider to find the best solution to the customer's needs.

Required Parameter Group

Output order ID

OUTPUT; CHAR(31)

The order identifier of the order log entry that was created or appended to.

Note: If only a configuration file was sent without any request information, this variable is set to blanks.

Input order ID

INPUT; CHAR(31)

When appending to a previous order assistance request, this is the order ID of the request to append to. If the order ID does not exist in the order database, a new request will be created with this order ID.

Note: If the input order ID is set to blanks, a new order assistance request will be created with a system-generated order ID.

The input order ID must be in the following format:

Displayable order ID CHAR(10)

The order ID as it is shown on the Work with Order Requests (WRKORDRQS) main display. **Note:** It is recommended that all characters in the displayable order ID be from the invariant character set.

Origin network address CHAR(21)
The network address of the system from which this order assistance request originated. The origin network address must be in the following format:

Type of network address CHAR(1)
Currently, only SNA addresses are supported by this API. Therefore, the only valid value for this field is 1.

Network address CHAR(20)

The SNA address must be in the following format:

Network ID CHAR(8)

Control Point CHAR(8)

Reserved CHAR(4)

The reserved field must be set to blanks.

Send or receive option

INPUT; BINARY(4)

Whether this order request is sent to the specified address, or was received from the specified address.

Values allowed for this field are:

0 The order assistance request is sent to the specified network address. If either requester or contact information are provided, this order entry will be appended to the order log of the local machine before being sent.

1 The order assistance request was received from the specified network address. An order log entry is appended to the order log of the local machine, showing that the request was received from the specified network address.

Contact information

INPUT; CHAR(*)

The name, phone number, fax number, and address that the service provider should use to contact the service requester. The customer number should also be associated with this request.

The contact information must be in the following format:

Number of variable length records BINARY(4)

Total number of all of the variable length records.

Variable length records	CHAR(*) The contact information attributes and their values. For more information on variable length records, see Format for Variable Length Record. For information on specific key values, see Contact Information Keys.

Note: The number of variable length records must be set to 0 if the user wants to do both of the following:

- Send ONLY a system configuration file.
- Not create an order log entry.

Configuration file

INPUT; CHAR(10)

The name of the system configuration file to be sent with this order assistance request.

The following special values are allowed:

**LOCAL*	A new system configuration file is created and sent with this order assistance request.
**NONE*	No system configuration file is sent with this order assistance request.

If the configuration file sent already exists on the machine this order assistance request is sent to, the file is overwritten by the new one.

Notes:

1. In order to be unique, the configuration file name is made up of the letter "Q" followed by:
   - The 2 character manufacture ID.
   - The 7 character serial number of the machine it was created on.
2. It is possible to send ONLY a configuration file without creating an order log entry. To do this, the values for the number of variable length records for the contact information and request information parameters must be set to 0.

Network address

INPUT; CHAR(*)

The network address to which this order assistance request should be sent, or the network address from which the order assistance request was received.

The network address must be in the following format:

Type of network address	BINARY(4) Currently, only SNA addresses are supported by this API. Therefore, the only valid value for this field is 1.
Length of network address	BINARY(4) The number of characters in the address.

Network address — CHAR(*)

The following special value is allowed:

**IBMSRV* — The order assistance request is sent to IBM through an ECS connection to the RETAIN(R) technical information network.

The SNA address must be in the following format:

Network ID	CHAR(8)
Control Point	CHAR(8)

Note: Any destination other than IBM must be running job QECS in order to successfully receive the order assistance request.

Request information

INPUT; CHAR(*)

Information about the order assistance request. This includes both a short and a detailed description of the request.

The information must be in the following format:

Number of variable length records	BINARY(4) Total number of all of the variable length records.
Variable length records	CHAR(*) The order request descriptions and their values. For more information on variable length records, see Format for Variable Length Record. For information on specific key values, see Request Information Keys.

Note: The number of variable length records must be set to 0 if the user wants to do both of the following:

- Send ONLY a system configuration file.
- Not create an order log entry.

System information

INPUT; CHAR(*)

Information about the system that this order assistance request originated on.

Number of variable length records	BINARY(4) Total number of all of the variable length records.

Variable length records	CHAR(*) The system information attributes and their values. For more information on variable length records, see Format for Variable Length Record. For information on specific key values, see System Information Keys.

Note: This variable is necessary if the user wants to specify system information for a system other than the one the API is executed on. Otherwise, the number of variable length records must be set to 0.

Error code

I/O; CHAR(*)

The structure in which to return error information. For the format of the structure, see Error Code Parameter.

Format for Variable Length Record

See Field Descriptions for descriptions of these fields.

Offset		Type	Field
Dec	Hex		
0	0	BINARY(4)	Length of variable length record
4	4	BINARY(4)	Attribute key
8	8	BINARY(4)	Length of data
12	C	CHAR(*)	Data

If the length of the data is longer then the key field's maximum data length, the data will be truncated. No message will be issued.

If the length of the data is smaller than the key identifier's data length, the data will be padded with blanks at the right. No message will be issued.

It is not an error to specify a key more than once. If duplicate keys are specified, the last specified value for that key is used.

Each variable length record must be 4-byte aligned. If not, unpredictable results may occur.

Field Descriptions

Attribute key. The attribute to be set.

Data. The repository for specific attribute information.

Length of data. The length of the value specified in the Data field.

Length of variable length record. The length of the entire variable length record, including this field.

Contact Information Keys

The following table describes the valid contact information keys. Keys 1 and 2 are required keys. If keys 3 through 12 are not specified, they default to blanks. See Key Descriptions for descriptions of these keys.

Key	Size	Description
1	CHAR(36)	Contact name
2	CHAR(30)	Telephone number
3	CHAR(30)	Fax number
4	CHAR(7)	Customer number
5	CHAR(36)	Company name
6	CHAR(48)	Street address line 1
7	CHAR(48)	Street address line 2
8	CHAR(48)	Street address line 3
9	CHAR(48)	City/State
10	CHAR(12)	Zip code
11	CHAR(20)	Country or region
12	CHAR(10)	User ID

Key Descriptions

City/state. The city and state the order should be sent to.

Note: The service provider receiving this request will only be able to display the first 36 characters of this field.

Company name. The name of the company as it appears on any orders sent for this request.

Contact name. The name of the person to contact about this order assistance request.

Note: The service provider receiving this request will only be able to display the first 28 characters of this field.

Country or region. The country or region to which the order should be sent.

Customer number. The customer number used for any orders generated by this request.

Fax number. The fax number of the person named in the Contact Name field.

Note: The service provider receiving this request will only be able to display the first 19 characters of this field.

Street address line 1. The first line in the street address this order should be sent to.

Note: The service provider receiving this request will only be able to display the first 36 characters of this field.

Street address line 2. The second line in the street address this order should be sent to.

Note: The service provider receiving this request will only be able to display the first 36 characters of this field.

Street address line 3. The third line in the street address this order should be sent to.

Note: The service provider receiving this request will only be able to display the first 36 characters of this field.

Telephone number. The telephone number of the person named in the Contact Name field.

Note: The service provider receiving this request will only be able to display the first 19 characters of this field.

User ID. The user ID of the person creating this append.

Note: If this parameter is not specified, the user ID defaults to the user ID that the job is running under.

Zip code. The zip code of where the order should be sent.

Request Information Keys

The following table describes the valid request information keys. If keys 1 and 2 are not specified, they default to blanks. See Key Descriptions for descriptions of these keys.

Key	Size	Description
1	CHAR(40)	Short description of assistance request
2	CHAR(714)	Detailed description of assistance request

Key Descriptions

Detailed description of assistance request. A full explanation of the order assistance request. The length of this field must be less than 714 characters.

Short description of assistance request. A summary of the order assistance request. The contents of this field will appear as the description in the order log. The length of this field must be less than 40 characters.

System Information Keys

The following table describes the valid system information keys. See Key Descriptions for descriptions of these keys.

Key	Size	Description
1	CHAR(10)	Serial number
2	CHAR(10)	System type
3	CHAR(10)	Model number
4	CHAR(2)	Country or region code
5	CHAR(3)	Language code
6	BINARY(4)	CCSID

Key Descriptions

CCSID. The CCSID the data is in.

If this key is not specified, the CCSID of the job sending this order assistance request is used.

Country or region code. The country or region code of the job that originated this order assistance request. If this key is not specified, the country or region code of the job sending this order assistance request will be used.

Language code. The language code of the job that originated this order assistance request.

If this key is not specified, the language code of the job sending this order assistance request will be used.

Model number. The model number of the machine that originated this order assistance request.

If this key is not specified, the model number of the machine sending this order assistance request is used.

Serial number. The serial number of the machine that originated this order assistance request. The serial number is made up of the 2-digit manufacture ID, followed by a dash, followed by the 7-digit sequential machine serial number.

If this key is not specified, the serial number of the machine sending this order assistance request will be used.

System type. The system type of the machine that originated this order assistance request.

If this key is not specified, the system type of the machine sending this order assistance request will be used.

Error Messages

Message ID	Error Message Text
CPF24B4 E	Severe error while addressing parameter list.
CPF3C90 E	Literal value cannot be changed.
CPF3CF1 E	Error code parameter not valid.
CPF3CF2 E	Error(s) occurred during running of &1 API.
CPF7D51 E	The input order ID parameter is a required parameter.
CPF7D52 E	&1 is not a valid value for the logging option parameter.
CPF7D53 E	Contact name and telephone number are required contact information fields.
CPF7D54 E	&1 is not a valid number of contact information fields.
CPF7D55 E	&1 is not a valid value for the contact information key.
CPF7D56 E	&1 is not a valid value for the configuration file parameter.
CPF7D57 E	The input order ID parameter is not valid.
CPF7D58 E	&1 is not a valid value for network address type.
CPF7D59 E	&1 is not a valid value for network address length.
CPF7D60 E	&1 is not a valid number of request information fields.

CPF7D61 E	&1 is not a valid request information key.
CPF7D62 E	&1 is not a valid number of system information fields.
CPF7D63 E	&1 is not a valid system information key.
CPF7D64 E	System configuration file not found.
CPF9872 E	Program or service program &1 in library &2 ended. Reason code &3.

API Introduced: V3R1

Top | Software Product APIs | APIs by category

Retrieve License Information (QLZARTV) API

Required Parameter Group:

1	Receiver variable	Output	Char(*)
2	Length of receiver variable	Input	Binary(4)
3	Format name for receiver variable	Input	Char(8)
4	Product identification	Input	Char(*)
5	Product identification format name	Input	Char(8)
6	Error code	I/O	Char(*)

Default Public Authority: *USE

Threadsafe: No

The Retrieve License Information (QLZARTV) API returns license information about a software product. The license information returned depends on the format specified.

- The LICR0100 format retrieves basic license information.
- The LICR0200 format retrieves basic license information, detailed license information, and the list of current license users.
- The LICR0300 format retrieves basic license information, detailed license information, the list of current license users, and additional information about the license users.

Authorities and Locks

Public API Authority

*USE

Required Parameter Group

Receiver variable

OUTPUT; CHAR(*)

The variable to receive the requested license information.

Length of receiver variable

INPUT; BINARY(4)

The length of the receiver variable provided. The length of receiver variable parameter may be specified up to the size of the receiver variable specified in the user program. If the length of receiver variable parameter specified is larger than the allocated size of the receiver variable specified in the user program, the results are not predictable. The minimum length is 8 bytes.

Format name for receiver variable

INPUT; CHAR(8)

The name of the format that identifies the type of license information to be retrieved.

The supported formats are:

LICR0100 Basic license information is retrieved. For more information, see LICR0100 Format.

LICR0200 Basic license information is retrieved along with detailed license information and license user information. For more information, see LICR0200 Format.

LICR0300 Basic license information is retrieved along with detailed license information and multiple-use license user information. For more information, see LICR0300 Format.

Product identification

INPUT; CHAR(*)

Information that uniquely identifies the product or feature whose license information will be retrieved. The structure of this information is determined by the name of the format.

Product identification format name

INPUT; CHAR(8)

The name of the format that describes the product identification.

The only format name supported is:

LICP0100 See LICP0100 Format.

Error code

I/O; CHAR(*)

The structure in which to return error information. For the format of the structure, see Error Code Parameter.

LICP0100 Format

The following table describes the format name supported for the format for product identification parameter. The format identifies the product or feature whose license information is to be retrieved. For detailed descriptions of the fields in the table, see Field Descriptions.

Offset		Type	Field
Dec	Hex		
0	0	CHAR(7)	Product ID
7	7	CHAR(6)	Release level
13	D	CHAR(4)	Feature

LICR0100 Format

The following describes the format of the license information returned in the receiver variable parameter. For detailed descriptions of the fields in the table, see Field Descriptions.

Offset		Type	Field
Dec	Hex		
0	0	BINARY(4)	Usage limit
4	4	BINARY(4)	Usage count
8	8	CHAR(2)	Usage type
10	A	CHAR(2)	Compliance type
12	C	CHAR(6)	License term
18	12	CHAR(6)	Release level

LICR0200 Format

The following describes the format of the license information returned in the receiver variable parameter. This format contains additional fields that LICR0100 does not have. For detailed descriptions of the fields in the table, see Field Descriptions.

Offset		Type	Field
Dec	Hex		
0	0	BINARY(4)	Bytes returned
4	4	BINARY(4)	Bytes available
8	8	BINARY(4)	Usage limit
12	C	BINARY(4)	Usage count
16	10	CHAR(2)	Usage type
18	12	CHAR(2)	Compliance type
20	14	CHAR(6)	License term
26	1A	CHAR(6)	Release level
32	20	BINARY(4)	Threshold value
36	24	BINARY(4)	Grace period
40	28	CHAR(7)	Date grace period expires
47	2F	CHAR(3)	Processor group
50	32	CHAR(2)	Reserved
52	34	BINARY(4)	Peak usage
56	38	CHAR(7)	Expiration date
63	3F	CHAR(8)	Vendor data
71	47	CHAR(16)	Product license information handle
87	57	CHAR(1)	Reserved
88	58	BINARY(4)	Offset to message queue list
92	5C	BINARY(4)	Number of message queues

96	60	BINARY(4)	Offset to current license user list
100	64	BINARY(4)	Number of current license users
104	68	BINARY(4)	Length of each current license user record
108	6C	BINARY(4)	Processor usage count
112	70	BINARY(4)	Processor peak usage count
116	74	CHAR(13)	Last peak date and time
129	81	CHAR(13)	Last update date and time
142	8E	CHAR(*)	Reserved
Note: Offsets vary depending on the number of message queues in the list.			
		CHAR(10)	Message queue name
		CHAR(10)	Message queue library name
Note: Offsets vary depending on the number of current license users and the length of each current license user record.			
		CHAR(*)	List of current license users

LICR0300 Format

The following describes the format of the license information returned in the receiver variable parameter. This format contains additional fields about license users that LICR0200 does not have. For detailed descriptions of the fields in the table, see Field Descriptions.

Offset		Type	Field
Dec	**Hex**		
0	0	BINARY(4)	Bytes returned
4	4	BINARY(4)	Bytes available
8	8	BINARY(4)	Usage limit
12	C	BINARY(4)	Usage count
16	10	CHAR(2)	Usage type
18	12	CHAR(2)	Compliance type
20	14	CHAR(6)	License term
26	1A	CHAR(6)	Release level
32	20	BINARY(4)	Threshold value
36	24	BINARY(4)	Grace period
40	28	CHAR(7)	Date grace period expires
47	2F	CHAR(3)	Processor group
50	32	CHAR(2)	Reserved
52	34	BINARY(4)	Peak usage
56	38	CHAR(7)	Expiration date
63	3F	CHAR(8)	Vendor data
71	47	CHAR(16)	Product license information handle
87	57	CHAR(1)	Reserved
88	58	BINARY(4)	Offset to message queue list

92	5C	BINARY(4)	Number of message queues
96	60	BINARY(4)	Offset to current license user list
100	64	BINARY(4)	Number of current license users
104	68	BINARY(4)	Length of each current license user record
108	6C	BINARY(4)	Length of each license user
112	70	BINARY(4)	Processor usage count
116	74	BINARY(4)	Processor peak usage count
120	78	CHAR(13)	Last peak date and time
133	85	CHAR(13)	Last update date and time
146	92	CHAR(*)	Reserved
Note: Offsets vary depending on the number of message queues in the list.			
		CHAR(10)	Message queue name
		CHAR(10)	Message queue library name
Note: Offsets vary depending on the number of current license users and the length of each current license user record.			
		BINARY(4)	Number of uses held
		CHAR(*)	License user
		CHAR(*)	Reserved

Field Descriptions

Bytes available.

The number of bytes of data available to be returned. All available data is returned if enough space is provided.

Bytes returned. The number of bytes of data returned.

Compliance type. The compliance type associated with this license. The compliance type determines the action taken when the value of the usage limit field is exceeded.

The valid values are:

01 The usage limit cannot be exceeded. The user who attempts to use the product or feature after the usage limit has been reached is prevented from accessing it. The product or feature cannot be accessed until the appropriate action is taken to increase the usage limit of it. A message indicating an attempt was made to exceed the usage limit is sent to each of the following:

- The QSYSOPR message queue
- The message queues specified on the Change License Information (CHGLICINF) command.

02 The user who attempts to use the product or feature after the usage limit has been reached is allowed access. A warning message indicating the usage limit is exceeded is sent to QSYSOPR and to the message queues specified on the Change License Information (CHGLICINF) command.

03 To use a product or feature with keyed compliance, the license must be installed using one of the following:

- A valid license key through the Add License Key Information (ADDLICKEY) command.
- The Add License Key Information (QLZAADDK) API.

This license key is provided by the software provider. The key ties the usage limit to the particular product or feature and to a particular system serial number. To change the usage limit, a user must get a new key from the software vendor. A user can use the product or feature after the usage limit is reached. However, the user is allowed access to the product or feature for only the number of days contained in the product's grace period. Once the grace period has expired, no users over the usage limit are able to use the product or feature until one of the following happens:

- A new license key is received from the software provider.
- The number of users fall below the usage limit.

A warning message indicating that the usage limit has been exceeded is sent to each of the following:

- The QSYSOPR message queue.
- The message queues specified on the CHGLICINF command.

Date grace period expires. The date that the grace period expires. Once a user has exceeded the usage limit, the date the grace period expires is set using the grace period and the current date. Before the grace period expires, a new license key needs to be obtained from the software vendor. If this is not done, users exceeding the usage limit are not allowed access to the product or feature.

9999999 This value indicates no grace period or that the grace period has expired.

CYYMMDD The date the grace period expires. C is the century, YY is the year, MM is the month, and DD is the day. The date must be numeric as follows:

- Century, where 0 indicates years 19*xx* and 1 indicates years 20*xx*.
- Month may not be greater than 12.
- Day may not be greater than 31.

Expiration date. The date the license will expire. The valid values are:

CYYMMDD C is the century, YY is the year, MM is the month, and DD is the day. The date must be numeric as follows:

- Century, where 0 indicates years 19*xx* and 1 indicates years 20*xx*.
- Month may not be greater than 12.
- Day may not be greater than 31.

9999999 The license does not have an expiration date.

Feature. The feature of the product. Valid values for the feature are 5001 through 9999.

Grace period. The number of days after a product first exceeds its usage limit that a user has to obtain a new license key. Before the grace period expires, a new license key needs to be obtained from the software vendor. If this is not done, users exceeding the usage limit are not allowed access to the product or feature. The date the grace period expires is calculated by adding the number of days in the grace period to the current date.

Last peak date and time. The date and time when the peak usage of the product or feature last occurred since the peak usage was reset to zero. In the CYYMMDDHHmmSS format as follows:

C Century, where 0 indicates years 19*xx* and 1 indicates years 20*xx*.

YY Year

MM Month

DD Day

HH Hour

mm Minute

SS Second

Last update date and time. The date and time when the usage limit was last updated. In the CYYMMDDHHmmSS format as follows:

C Century, where 0 indicates years 19*xx* and 1 indicates years 20*xx*.

YY Year

MM Month

DD Day

HH Hour

mm Minute

SS Second

Length of each current license user record. The length of each current license user record. This is the length of the license user plus the length of any additional license user information.

Length of each license user. The length of each license user. This is the same value that is used during the request and release of this product. This may be a value of 1 through 80.

License term. The extent of time the authorized usage limit for a product lasts. Each time a new license term is installed for a product, the authorized usage limit must be set by doing each of the following:

- Obtaining a new license key.
- Using the Add License Key Information (ADDLICKEY) command.

Possible values are:

Vx The authorized usage limit is valid only for the entire version of the product or feature.

VxRy The authorized usage limit is valid only for the entire release of the product or feature.

VxRyMz The authorized usage limit is valid only for the modification level of the product.

Where the x and y can be a number from 0 through 9. Z can be a number 0 through 9 or a letter A through Z.

License user. A user that currently holds one or more uses of the product or feature.

List of current license users. A list of all the current users of the product.

Message queue library name. The library where the message queue resides.

Message queue name. The name of message queue.

Number of current license users. The number of current license users in the list.

Number of message queues. The number of message queues in the list.

Number of uses held. The number of license uses held by this license user.

Offset to current license user list. The offset from the beginning of the receiver variable to the start of the first current license user. This offset is 0 if there are no license users or if the size of the receiver variable is not large enough to hold any license users.

Offset to message queue list. The offset from the beginning of the receiver variable to the start of the first message queue name and library. This offset is 0 if there are no message queues or if the size of the receiver variable is not large enough to hold any message queues.

Peak usage. The maximum number of license users that have accessed the product or feature at one time. The peak usage may be reset using option 10 of the Work License Information (WRKLICINF) command. If the product is using processor usage type, the peak usage value will be rounded up to the next whole number. For instance, an actual peak usage of 2.15 would round up to 3. See the processor peak usage count field for the processor peak usage count in hundreths of a processor.

Processor group. The processor group of this system. A processor group is the grouping of system model numbers by relative processor size.

Processor usage count The processor usage count is the number of hundreths of processors in the logical partition configured at the time the product was used. This field is set to 0 for products that do not have a processor usage type.

Processor peak usage count The maximum processor usage count in hundreths of processors. The processor peak usage may be reset using option 10 of the Work License Information (WRKLICINF) command. This field is set to 0 for products that do not have a processor usage type.

Product ID. The product ID of the product or feature whose license information is to be retrieved.

Product license information handle. The product information handle is passed back. It may be used within the application to verify that the product attributes are the same as the original license information created by the software provider. This handle will not be stored and will be generated each time license information is retrieved.

Release level. The version, release, and modification level of the product whose license information was requested. This is returned in the receiver variable parameter. If you specified *ONLY in the release field of the LICP0100 format, the actual release level is returned here.

Reserved. If this field is input, character fields must be set to blanks and binary fields must be set to hexadecimal zeros.

Threshold value. The threshold for this product or feature.

The threshold indicates you want a message sent to the system operator message queue stating that a product or feature is reaching the usage limit.

-1	The threshold value is *NOMAX.
0-999999	The threshold value.

Usage count. The usage count for the product or feature at the time of the retrieve operation. Valid values are 0 through 999999. If the product is using processor usage type, the usage count value will be rounded up to the next whole number. For instance, an actual usage count of 2.15 would round up to 3. See the processor usage count field for the processor usage count in hundreths of a processor.

Usage limit. The usage limit for this license.

-1 Any number of users are allowed to access the product or feature.

0-999999 The number of users allowed to access the product.

Usage type. The usage type associated with this license.

The valid values are:

01 The usage type is concurrent. It is for the number of unique jobs accessing the product at one time.

02 The usage type is registered. It is for the number of unique license users registered by the product.

03 The license usage is by processors. Counts the number of processors that are assigned to the logical partition.

Vendor data. Information the vendor defined at Generate License Key time.

Error Messages

Message ID	Error Message Text
CPF0C1C E	Release level &1 not valid.
CPF0C1E E	Error occurred during running of &1 API.
CPF2206 E	User needs authority to do requested function on object.
CPF2207 E	Not authorized to use object &1 in library &3 type *&2.
CPF24B4 E	Severe error while addressing parameter list.
CPF3CF1 E	Error code parameter not valid.
CPF3C21 E	Format name &1 is not valid.
CPF3C24 E	Length of the receiver variable is not valid.
CPF3C90 E	Literal value cannot be changed.
CPF9E11 E	License information not retrieved.
CPF9E13 E	More than one release found.
CPF9E15 E	Error in license management function.
CPF9872 E	Program or service program &1 in library &2 ended. Reason code &3.

API Introduced: V2R3

Top | Software Product APIs | APIs by category

Retrieve License Key Information (QLZARTVK) API

Required Parameter Group:

1	Receiver variable	Output	Char(*)
2	Length of receiver variable	Input	Binary(4)
3	Format name of receiver variable	Input	Char(8)
4	Product identification	Input	Char(*)
5	Product identification format name	Input	Char(8)
6	System	Input	Char(*)
7	System format name	Input	Char(8)
8	Error code	I/O	Char(*)

Default Public Authority: *USE

Threadsafe: No

The Retrieve License Key Information (QLZARTVK) API retrieves the license key information for the specified systems from the license repository. The information retrieved is about the specified product, license terms, and features. The license repository is used to store license key information. It contains a record for each license key for every unique product, license term, feature, and system. The repository may contain licenses for any system, and the product does not need to be installed.

Authorities and Locks

API QLZARTVK Authority

*PUBLIC(*EXCLUDE)

Required Parameter Group

Receiver variable

OUTPUT; CHAR(*)

The receiver variable that receives the information requested. You can specify the size of the area to be smaller than the format requested as long as you specify the length parameter correctly. As a result, the API returns only the data that the area can hold.

Length of receiver variable

INPUT; BINARY(4)

The length of the receiver variable provided. The length of receiver variable parameter may be specified up to the size of the receiver variable specified in the user program. If the length of receiver variable parameter specified is larger than the allocated size of the receiver variable specified in the user

program, the results are not predictable. The minimum length is 8 bytes.

Format name for receiver variable

INPUT; CHAR(8)

The name of the format that identifies the type of license information to be retrieved.

The only format name supported is:

LICV0100 Basic license information is retrieved. For more information, see LICV0100 Format.

Product identification

INPUT; CHAR(*)

Information that uniquely identifies the product or feature whose license information is to be retrieved. The structure of this information is determined by the name of the format.

Product identification format name

INPUT; CHAR(8)

The name of the format that describes the product identification.

The only format name supported is:

LICT0100 See LICT0100 Format.

System

INPUT; CHAR(*)

This indicates the system serial number for which licenses will be retrieved. The structure of this information is determined by the name of the format.

System format name

INPUT; CHAR(8)

The name of the format containing the systems.

The only format name supported is:

LICS0100 The systems used as input to the API. For details, see the LICS0100 Format.

Error code

I/O; CHAR(*)

The structure in which to return error information. For the format of the structure, see Error Code Parameter.

LICV0100 Format

The following table describes the format of the license key information returned in the receiver variable parameter. For a detailed description of the fields in this table, see Field Descriptions.

Offset		Type	Field
Dec	Hex		
0	0	BINARY(4)	Bytes returned
4	4	BINARY(4)	Bytes available
8	8	BINARY(4)	Offset to license key records
12	C	BINARY(4)	Number of license key records in list
16	10	BINARY(4)	Length of license key record
20	14	CHAR(*)	Reserved
Note: The offsets of the following fields vary. These fields repeat, in the order listed, for each specified product, license term, feature, and system.			
		CHAR(7)	Product identifier
		CHAR(6)	License term
		CHAR(4)	Feature
		CHAR(8)	System serial number
		CHAR(4)	Processor group
		CHAR(3)	Reserved
		BINARY(4)	Usage limit
		CHAR(7)	Expiration date
		CHAR(8)	Vendor data
		CHAR(18)	License key
		CHAR(15)	Reserved
		CHAR(*)	Reserved

LICT0100 Format

The following information uniquely describes the product or feature whose license information is to be added. For a detailed description of the fields in this table, see Field Descriptions.

Offset		Type	Field
Dec	Hex		
0	0	CHAR(7)	Product identifier
7	7	CHAR(6)	License term
13	D	CHAR(4)	Feature

LICS0100 Format

The following information describes the systems for which the license information is to be retrieved. For a detailed description of the field, see Field Descriptions.

Offset		Type	Field
Dec	Hex		
0	0	CHAR(8)	System serial number

Field Descriptions

Bytes available. The number of bytes of data available to be returned. All available data is returned if enough space is provided.

Bytes returned. The number of bytes of data returned.

Expiration date. The date the license will expire.

The valid values are:

CYYMMDD C is the century, YY is the year, MM is the month, and DD is the day. The date must be numeric as follows:

- Century, where 0 indicates years 19*xx* and 1 indicates years 20*xx*.
- Month may not be greater than 12.
- Day may not be greater than 31.

9999999 The license does not have an expiration date.

Feature. The feature of the product. Valid values for the feature are 5001 through 9999.

For the input parameter the valid special value is:

**ALL* The license key information for all the features will be retrieved.

Length of license key record. The length of each license key record.

License key. The license key for the product, license term, feature, and system.

License term. The license term of the product.

Possible values are:

Vx The authorized usage limit is valid only for the entire version of the product or feature.

VxRy The authorized usage limit is valid only for the entire release of the product or feature.

VxRyMz The authorized usage limit is valid only for the modification level of the product.

Where the x and y can be a number from 0 through 9. Z can be a number 0 through 9 or a letter A through Z.

For the input parameter the valid special value is:

**ALL* The license key information for all license terms will be retrieved.

Number of license key records. The number of license key records in the list.

Offset to license key records. The offset from the beginning of the receiver variable to the first license key record.

Processor group. The processor group for which this license is for. This field is left justified.

The valid values are:

**ANY* The license key is valid for any processor group.

Product identifier. The product identifier of the product or feature.

For the input parameter the valid special value is:

**ALL* The license key information for all the product identifiers will be retrieved.

Reserved. Reserved for future use. If this field is input, character fields must be set to blanks and binary fields must be set to hexadecimal zeros.

System serial number. The system serial number.

For the input parameter, the valid special values are:

**ALL* License key information for all systems will be retrieved.

**LOCAL* License key information for only this local system will be retrieved.

**REMOTE* License key information for all systems except this local system will be retrieved.

The special values are left justified. A value other than a special value must be right justified.

Usage limit. The usage limit for this license.

-1 There is no maximum number of license users for this product.

0-999999 The maximum number of license users for this product.

Vendor data. The data for this field comes from the software provider along with the license key information.

Error Messages

Message ID	Error Message Text
CPF24B4 E	Severe error while addressing parameter list.
CPF3CF1 E	Error code parameter not valid.
CPF3C21 E	Format name &1 is not valid.

CPF3C90 E	Literal value cannot be changed.
CPF9E15 E	Error in license management function.
CPF9E54 E	License term &1 not valid.
CPF9E58 E	License key information not found.
CPF9E6D E	Feature &3 not valid.
CPF9822 E	Not authorized to file &1 in library &2.
CPF9826 E	Cannot allocate file &2.
CPF9872 E	Program or service program &1 in library &2 ended. Reason code &3.

API Introduced: V3R1

Top | Software Product APIs | APIs by category

Retrieve Product Information (QSZRTVPR) API

Required Parameter Group:

1	Receiver variable	Output	Char(*)
2	Length of receiver variable	Input	Binary(4)
3	Format name	Input	Char(8)
4	Product information	Input	Char(*)
5	Error code	I/O	Char(*)

Optional Parameter:

6	Product information format name	Input	Char(8)

Default Public Authority: *USE

Threadsafe: Yes

The Retrieve Product Information (QSZRTVPR) API returns information about a software product. The information is requested by specifying a product ID, release level, option number, and load ID; not by specifying an object name. The Display Software Resources (DSPSFWRSC) command and the Select Product (QSZSLTPR) API will obtain a list of installed products about which you can retrieve information.

You can use this API to:

- Retrieve general information about a product load, including whether the product load is installed or not.
- Retrieve the library list of a product load.
- Retrieve the folder list of a product load.
- Retrieve the object list of a product load.
- Retrieve the directory list of a product load.
- Retrieve the list of option and load ID pairs that are valid for a product ID and release combination. This is based on what is listed in the product definition (*PRDDFN) for that product ID and release combination.
- Retrieve information from a product definition, including:
  - The copyright information.
  - The release date.
  - The message file name and library.
  - Whether the product allows multiple releases.
  - The message ID for each option.
  - Whether each option allows dynamic naming.
- Retrieve the current release level of the operating system.
- Retrieve the previous release level of the operating system.
- Retrieve a list of valid release levels of the operating system from a given release level through the

currently installed release level.

- Retrieve the primary language ID of a product.

Note: The Retrieve Object Description (QUSROBJD) API can be used to retrieve product information from the object description of an object. The product ID and release level from the object description is returned by QUSROBJD in format OBJD0300.

Authorities and Locks

Product Availability Authority

None

Product Availability Lock

*SHRRD

The product availability object resides in the QUSRSYS library.

Product Definition Authority

None

Product Load Authority

None

Required Parameter Group

Receiver variable

OUTPUT; CHAR(*)

The variable to receive the requested information.

Length of receiver variable

INPUT; BINARY(4)

The length of the receiver variable in bytes. The value specified must be at least 8.

Format name

INPUT; CHAR(8)

The content and format of the information returned.

The possible format names are:

PRDR0100	Returns basic information about the product load. For more information, see PRDR0100 Format.
PRDR0200	Returns a list of the principal and additional libraries for this product load, along with the basic information. Exit program names and other related information are also returned. For more information, see PRDR0200 Format.

PRDR0300 Returns a list of the folders for this product load, along with the basic information. For more information, see PRDR0300 Format.

PRDR0400 Returns a list of the packaged objects for this product load, along with the basic information. For more information, see PRDR0400 Format.

PRDR0500 Returns the information that was entered when the product definition object was created. This includes a record for each option listed in the product definition (*PRDDFN) for that product ID and release level. For more information, see PRDR0500 Format.

PRDR0600 Returns a list of option and load ID pairs that are valid for the specified product ID and release level. This is based on what is listed in the product definition (*PRDDFN) for that product ID and release. For more information, see PRDR0600 Format.

PRDR0700 Returns a list of release levels of the operating system. The list starts with the release level passed in by the caller and includes all releases of the operating system through the currently installed release. For more information, see PRDR0700 Format.

PRDR0800 Returns a list of product home directories and product directories for this product load, along with the basic information. For more information, see PRDR0800 Format.

Product information

INPUT; CHAR(*)

The structure that contains values for which product information is to be retrieved. The structure provided depends on which product information format is requested. For more information, see Product Information Format.

Error code

I/O; CHAR(*)

The structure in which to return error information. For the format of the structure, see Error Code Parameter.

Optional Parameter

Product information format name

INPUT; CHAR(8)

The content and format of the product information parameter.

The possible format names follow:

PRDI0100 The product ID, release level, product option, and load ID. This is the default value when this parameter is not present. For more information, see PRDI0100 Format.

PRDI0200 Everything in format PRDI0100 plus the CCSID that the directory names for format PRDR0800 are to be returned in. For more information, see PRDI0200 Format.

Product Information Format

Information passed in the product information parameter can be in one of the following formats. For detailed descriptions of each field, see Field Descriptions.

PRDI0100 Format

Offset		Type	Field
Dec	Hex		
0	0	CHAR(7)	Product ID
7	7	CHAR(6)	Release level
13	D	CHAR(4)	Product option
17	11	CHAR(10)	Load ID

PRDI0200 Format

Offset		Type	Field
Dec	Hex		
0	0	CHAR(7)	Product ID
7	7	CHAR(6)	Release level
13	D	CHAR(4)	Product option
17	11	CHAR(10)	Load ID
27	1B	CHAR(1)	Reserved
28	1C	BINARY(4)	Length of product information parameter
32	20	BINARY(4)	CCSID for returned directory names
36	24	CHAR(*)	Reserved

Field Descriptions

CCSID for returned directory names. The CCSID in which the directory names should be returned. If this field is zero or 65535, the directories are returned in the job default CCSID.

Length of product information parameter. The total number of bytes specified on the product information parameter. The value specified must be from 36 through 256.

Load ID. The load ID for which information is being requested. Load IDs are 4 characters in length; for example, 2924 is the load ID for an English national language version (NLV).

You can use this special value for the load ID:

**CODE* The load ID of the code load for the given product ID, release level, and option.

Product ID. The product ID for which information is being requested.

You can use this special value for the product ID:

**OPSYS* The product ID for the operating system for the specified release level. The product ID depends on the release level specified.

Product option. The option number for which information is being requested. Use 0000 for the base option. Valid values are 0000 through 0099, where each character is a digit.

Release level. The release level for which information is being requested. The release level must be a valid special value, or the release level must be in the format VxRxMy. Valid values for x are 0 through 9. Valid values for y are 0 through 9 and A through Z.

You can use these special values for the release level:

**CUR* Uses the release level of the currently installed operating system.

**ONLY* Uses the only release level for which a product load (*PRDLOD) is found. If loads are found for multiple release levels, an error (CPF0C30) will occur.

**PRV* Uses the previous release with modification level 0 of the operating system.

Examples follow:

Example 1: If the current release level is V2R1M0, specifying *PRV for the release level parameter returns V1R3M0.

Example 2: If the current release level is V2R1M1, specifying *PRV for the release level parameter returns V1R3M0, not V2R1M0.

Example 3: If the current release level is V2R2M0, specifying *PRV for the release level parameter returns V2R1M0.

Note: In these examples, the current release level values are used to show how *PRV is determined; this API did not exist before V2R3M0.

Reserved. This field must contain hexadecimal zeros.

Format of the Returned Information

Information returned in the receiver variable parameter can be in one of the following formats. For detailed descriptions of the fields for each format, see Field Descriptions.

PRDR0100 Format

If the product load is not known to the system, an error (CPF0C1F) will occur.

Offset		

Dec	Hex	Type	Field
0	0	BINARY(4)	Bytes returned
4	4	BINARY(4)	Bytes available
8	8	BINARY(4)	Reserved
12	C	CHAR(7)	Product ID
19	13	CHAR(6)	Release level
25	19	CHAR(4)	Product option
29	1D	CHAR(4)	Load ID
33	21	CHAR(10)	Load type
43	2B	CHAR(10)	Symbolic load state
53	35	CHAR(10)	Load error indicator
63	3F	CHAR(2)	Load state
65	41	CHAR(1)	Supported flag
66	42	CHAR(2)	Registration type
68	44	CHAR(14)	Registration value
82	52	CHAR(2)	Reserved
84	54	BINARY(4)	Offset to additional information
88	58	CHAR(4)	Primary language load identifier
92	5C	CHAR(6)	Minimum target release
98	62	CHAR(6)	Minimum VRM of *BASE required by option
104	68	CHAR(1)	Requirements met between base and option value
105	69	CHAR(3)	Level
108	6C	CHAR(*)	Reserved

PRDR0200 Format

If the *PRDLOD object does not exist, an error (CPF0C1F) will occur.

The fields following the library records field define that array. The number of entries in the array is the number of primary libraries for this load.

Offset		Type	Field
Dec	Hex		
0	0	CHAR(*)	Everything from format PRDR0100
The decimal and hexadecimal offsets are determined by using the value in the offset to additional information field of format PRDR0100.		CHAR(10)	Secondary language library name
		CHAR(2)	Reserved
		BINARY(4)	Number of primary libraries
		BINARY(4)	Offset to library records
		CHAR(*)	Reserved
		ARRAY of CHAR(*)	Library records

The decimal and hexadecimal offsets are determined by using the value in the offset to library records field and the offset to next library record field.	BINARY(4)	Offset to next library record
	CHAR(10)	Primary library name
	CHAR(10)	Installed library name
	CHAR(10)	Library type
	CHAR(10)	Library authority
	CHAR(10)	Library create authority
	CHAR(10)	Postoperation exit program name
	BINARY(4)	Number of preoperation exit program names
	ARRAY of CHAR(10)	Preoperation exit program names
	CHAR(*)	Reserved

PRDR0300 Format

If the *PRDLOD object does not exist, an error (CPF0C1F) will occur.

The fields following the folder records field define an entry in that array. The number of entries in the array is the number of primary folders for this load.

Offset		Type	Field
Dec	**Hex**		
0	0	CHAR(*)	Everything from format PRDR0100
The decimal and hexadecimal offsets are determined by using the value in the offset to additional information field of format PRDR0100.		BINARY(4)	Number of primary folders
		BINARY(4)	Length of one folder record
		BINARY(4)	Offset to folder records
		CHAR(*)	Reserved
		ARRAY of CHAR(*)	Folder records
The decimal and hexadecimal offsets are determined by using the value in the offset to folder records field and the length of one folder record field.		CHAR(63)	Primary folder
		CHAR(63)	Installed folder
		CHAR(*)	Reserved

PRDR0400 Format

If the product load has not been packaged, an error (CPF0C1F) will occur. There may have been PTF activity for this product load. If so, this list might not contain all the objects that would be saved by the Save Licensed Program (SAVLICPGM) command. Error CPF0C1B is returned if this format is requested for the base option of the operating system.

The fields following the object records field define an entry in that array. The number of entries in the array is the number of objects packaged for this load.

Offset		Type	Field
Dec	Hex		
0	0	CHAR(*)	Everything from format PRDR0100
The decimal and hexadecimal offsets are determined by using the value in the offset to additional information field of format PRDR0100.		BINARY(4)	Number of objects
		BINARY(4)	Length of one object record
		BINARY(4)	Offset to object records
		CHAR(*)	Reserved
		ARRAY of CHAR(*)	Object records
The decimal and hexadecimal offsets are determined by using the value in the offset to object records field and the length of one object record field.		CHAR(10)	Object name
		CHAR(10)	Installed library name
		CHAR(10)	Object type
		CHAR(*)	Reserved

PRDR0500 Format

If the product definition for the specified product ID and release level does not exist, an error (CPF0C1F) will occur. Error CPF0C1B is returned if this format is requested without specifying product option 0000 and load ID *CODE. Product option 0000 must be specified for this format even though the information returned is then for all options.

The fields following the option records field define an entry in that array. The number of entries in the array is the number of product options for this product and release.

Offset		Type	Field
Dec	Hex		
0	0	CHAR(*)	Everything from format PRDR0100
The decimal and hexadecimal offsets are determined by		CHAR(1)	Allow multiple releases
		CHAR(1)	Release date century
		CHAR(6)	Release date

using the value in the offset to additional information field of format PRDR0100.	CHAR(4)	Copyright first year
	CHAR(4)	Copyright current year
	CHAR(10)	Message file object name
	CHAR(10)	Message file library name
	BINARY(4)	Number of option records
	BINARY(4)	Length of one option record
	BINARY(4)	Offset to option records
	CHAR(1)	Allow mixed releases
	CHAR(*)	Reserved
	ARRAY of CHAR(*)	Option records
The decimal and hexadecimal offsets are determined by using the value in the offset to option records field and the length of one option record field.	CHAR(4)	Product option
	CHAR(1)	Allow dynamic naming
	CHAR(7)	Product option message ID
	CHAR(6)	Minimum required VRM of option
	CHAR(*)	Reserved

PRDR0600 Format

Error CPF0C1F occurs if the product definition does not exist. Error CPF0C1B is returned if this format is requested without specifying product option 0000 and load ID *CODE.

The fields following the load records field define an entry in that array. The number of entries in the array is the number of loads for this product and release.

Offset		Type	Field
Dec	Hex		
0	0	CHAR(*)	Everything from format PRDR0100
The decimal and hexadecimal offsets are determined by using the value in the offset to additional information field of format PRDR0100.		BINARY(4)	Number of load records
		BINARY(4)	Length of one load record
		BINARY(4)	Offset to load records
		CHAR(*)	Reserved
		ARRAY of CHAR(*)	Load records
The decimal and hexadecimal offsets are determined by		CHAR(4)	Product option

using the value in the offset to load records field and the length of one load record field.	CHAR(4)	Load ID
	CHAR(*)	Reserved

PRDR0700 Format

When this format is requested, valid values for the release level field are V1R3M0 and all release levels for which the operating system was made available through the currently installed release level of the operating system.

If the release level field is not a valid value, an error (CPF0C1C) will occur. Error CPF0C1B is returned if this format is requested without specifying the product option as 0000, product ID as *OPSYS, and load ID as *CODE.

Offset		Type	Field
Dec	**Hex**		
0	0	BINARY(4)	Bytes returned
04	04	BINARY(4)	Bytes available
08	08	BINARY(4)	Number of releases returned
12	0C	CHAR(4)	Reserved
16	10	ARRAY of CHAR(6)	Release level (for each returned operating system release level)

PRDR0800 Format

If the *PRDLOD object does not exist, an error (CPF0C1F) will occur.

The product home directory is a grouping mechanism. It is designed to be the parent directory for several product directory paths. The directory information array will contain an entry for each primary full path for the load. This entry will have offsets to the primary and installed path names and the public object authorities for the directory.

All offsets within this structure will be set to 0 when the offset would be beyond the end of the receiver variable.

The following information is included to help clarify the use of format PRDR0800.

The primary full path being used in this example is */QSom/Class*. The primary product home directory portion of this path is */QSom*. The product directory is *Class*. These three pieces of information are returned as character array */QSom/Class*.

For example, if */QSom/Class* were 1000 bytes from the beginning of the receiver variable, the offset to primary full path name field would be 1000. To find the primary product home directory name, an offset of 1000 would be used as well. The length of primary full path name field would be 11. The length of primary product home directory name field would be 5. The offset to primary product directory name field would be 1006. The length

of the primary product directory name would be: the length of primary full path name field plus the offset to primary full path name field minus the offset to primary product directory name field (11 + 1000 - 1006 = 5 bytes).

Each product directory associated with a product load will have a directory information array entry. This directory information array entry contains the information to access the different parts of the character string explained above.

Offset		Type	Field
Dec	Hex		
0	0	CHAR(*)	Everything from format PRDR0100
The decimal and hexadecimal offsets are determined by using the value in the offset to additional information field of format PRDR0100.		BINARY(4)	Number of primary full paths
		BINARY(4)	Length of one directory information entry
		BINARY(4)	Offset to directory information array
		BINARY(4)	CCSID of returned directories
		CHAR(1)	Error indicator on CCSID conversion
		CHAR(*)	Reserved
The decimal and hexadecimal offsets are determined by using the value in the offset to directory information array.		ARRAY of CHAR(*)	Directory information array
		BINARY(4)	Length of primary full path name
		BINARY(4)	Offset to primary full path name
		BINARY(4)	Length of primary product home directory name
		BINARY(4)	Offset to primary product directory name
		BINARY(4)	Length of installed full path name
		BINARY(4)	Offset to installed full path name
		BINARY(4)	Length of installed product home directory name
		BINARY(4)	Offset to installed product directory name
		BINARY(4)	Number of public object authorities available
		BINARY(4)	Offset to public object authority array
		CHAR(10)	Public data authority
		CHAR(*)	Reserved
The decimal and hexadecimal offsets are determined by using the value in the offset to primary full path name.		CHAR(*)	Primary full path name (contains the primary product home directory name and the primary product directory name)

The decimal and hexadecimal offsets are determined by using the value in the offset to installed full path name.	CHAR(*)	Installed full path name (contains the installed product home directory name and the installed product directory name)
The decimal and hexadecimal offsets are determined by using the value in the offset to public object authority array.	ARRAY of CHAR(10)	Public object authorities array

Field Descriptions

Allow dynamic naming. Whether the names of product libraries and root folders for this product option can be dynamically changed without causing a product error.

Possible values are:

0 Cannot be dynamically named.

1 Can be dynamically named.

Allow multiple releases. Whether this product can be installed at a release level different from the current release level without installing over the current release.

Possible values are:

0 This product cannot be installed at a release level different from the current release level without installing over the current release.

1 This product can be installed at a release level different from the current release level without installing over the current release.

Allow mixed releases. Whether this product allows mixed releases between its *BASE and options.

Possible values are:

0 The *BASE option and other options of this product cannot be at different release levels.

1 The *BASE option and other options of this product can be at different release levels.

Bytes available. The number of bytes of data available to be returned to the user.

Bytes returned. The number of bytes returned to the user. This is the lesser of the number of bytes available and the length of the receiver variable.

CCSID of returned directories. The value of the CCSID in which the directories were returned. This will be the requested CCSID if the error indicator on CCSID conversion field is 0.

Copyright current year. The value specified for the copyright current year when the product definition for this product load was created. If no copyright current year was specified when the product definition was created, the copyright current year is blank.

Copyright first year. The value specified for the copyright first year when the product definition for this product load was created. If no copyright first year was specified when the product definition was created, the copyright first year is blank.

Directory information array. An array that contains an element for each primary full path. The length of an element is specified by length of one directory information entry. The number of elements is the number of primary full paths.

Error indicator on CCSID conversion. Whether the CCSID conversion to the requested CCSID was successful. If the requested CCSID conversion fails, the CCSID in which the directories are returned is identified in the CCSID of returned directories field.

Possible values follow:

0 CCSID conversion was successful

1 CCSID conversion failed

Folder records. An array in which each entry includes the primary folder, installed folder, and reserved fields.

Installed folder. This can be one of the following:

- The name of the folder specified as the development folder when the load was created.
- The name given to the primary folder when the product was installed.

Installed full path name. The installed full path name for the associated primary full path. It contains a directory name in the CCSID that is indicated in the CCSID of returned directories field. The installed full path contains the installed product home directory concatenated with a slash (/), which is concatenated with the installed product directory name. The length of the product directory is the length of the installed full path plus the offset to the installed full path minus the offset to the installed product directory name.

Installed library name. For a library record, this can be one of the following:

- The name of the library specified as the development library when the load was created.
- The name given to the primary library when the product was installed.

For an object record, the name of the library where the object should exist. The object might not exist in this library if the object had been deleted, or the library had been renamed.

Length of installed full path name. The number of bytes in this installed full path name.

Length of installed product home directory name. The number of bytes in the installed product home directory.

Length of one directory information entry. The number of bytes in each directory information array entry.

Length of one folder record. The number of bytes in each folder record.

Length of one load record. The number of bytes in each load record.

Length of one object record. The number of bytes in each object record.

Length of one option record. The number of bytes in each option record.

Length of primary full path name. The number of bytes in this primary full path name.

Length of primary product home directory name. The number of bytes in the primary product home directory.

Level. The level identifier of the product for which information was returned. The format is Lxx. The returned value is blank for all products other than the operating system and Licensed Internal Code.

Library authority. The public authority given to the library by the Restore Licensed Program (RSTLICPGM) command when this load is installed if the library does not exist. This field will be blank if the product load has not been successfully packaged.

Possible values are:

**ALL*	The library is created with the public authority set to *ALL.
**USE*	The library is created with the public authority set to *USE.
**CHANGE*	The library is created with the public authority set to *CHANGE.
**EXCLUDE*	The library is created with the public authority set to *EXCLUDE.

Library create authority. The create authority set for this library by the Restore Licensed Program (RSTLICPGM) command when this load is installed if the library does not exist. This field will be blank if the product load has not been successfully packaged.

Possible values are:

**ALL*	The library is created with the public authority set to *ALL.
**USE*	The library is created with the public authority set to *USE.
**CHANGE*	The library is created with the public authority set to *CHANGE.
**EXCLUDE*	The library is created with the public authority set to *EXCLUDE.

Library records. An array in which each entry includes the following fields:

- Offset to next library record
- Primary library name
- Installed library
- Library type
- Library authority
- Library create authority
- Postoperation exit program name
- Number of preoperation exit program names
- Preoperation exit program names
- Reserved

Library type. The type of library created by the Restore Licensed Program (RSTLICPGM) command when this

load is installed if the library does not exist. This field will be blank if the product load has not been successfully packaged.

Possible values are:

**PROD* The library created is a production library.

**TEST* The library created is a test library.

Load error indicator. Whether there is a known error for this load.

The possible values are:

**ERROR* An error was found the last time that the state of this load was checked or updated. For example, a restore, delete, or save licensed program function might be in progress or might not have completed. The state of a load can be checked using the Check Product Option (CHKPRDOPT) command. See the Control Language (CL) information in the iSeries Information Center for information on the CHKPRDOPT command.

**NONE* No error was found the last time that the state of this load was checked or updated.

Note: This does not mean that the product is necessarily installed. Refer to the symbolic load state field to determine if the load is installed or not.

Load ID. The load ID of the product load for which information was returned. For the load records, the load ID field returns the load IDs that have been specified when the product definition was created.

Load records. An array for which each entry includes the product option, load ID, and a reserved field.

Load state. The state of the load for which information was returned.

The possible values are:

10 The load is defined. The product load object for this load does not exist. When a product definition is created, a code load is defined for each product option, and language loads can be defined.

20 The product load object for this load exists. Before it can be saved using the Save Licensed Program (SAVLICPGM) command, it must be packaged with one of the following:

- The Package Product Option (PKGPRDOPT) command.
- The Package Product Option (QSZPKGPO) API.

3E A Restore Licensed Program (RSTLICPGM) command did not complete successfully. A preoperation exit program failed. The product being replaced had been packaged, but not installed.

3F A RSTLICPGM command failed. A preoperation exit program did not fail. The product being replaced had been packaged, but not installed.

30 The product load object for this load has been packaged with the PKGPRDOPT command or the QSZPKGPO API.

32 The product load object for this load has been packaged with the PKGPRDOPT command or the QSZPKGPO API. One of the following occurred:

- A development library or folder was renamed, but the product does not allow dynamic naming. (A product specifies whether or not it allows dynamic naming when the product definition object is created.)
- The product definition or product load for a packaged load was renamed or moved to another library.

33 The product load object for this load has been packaged with the PKGPRDOPT command or the QSZPKGPO API. However, an object was found to be damaged the last time that the CHKPRDOPT command or SAVLICPGM command was used for this load.

34 The product load object for this load has been packaged with the PKGPRDOPT command or the QSZPKGPO API. One of the following occurred:

- An attempt was made to delete the product load using the delete licensed program function and the function failed.
- A packaged object was missing the last time that the CHKPRDOPT command or SAVLICPGM command was used for this load.

35 A RSTLICPGM command is in progress. The product being replaced had been packaged, but not installed.

38 A Delete Licensed Program (DLTLICPGM) command is in progress. The product being deleted had been packaged, but not installed.

50 A RSTLICPGM command is in progress. The product being replaced had been installed.

53 A DLTLICPGM command is in progress. The product being deleted had been installed.

59 This product is an IBM-supplied product, and it is not compatible with the currently installed release level of the operating system. An error occurred when the product was restored or when the operating system was installed. The IBM-supplied product is at a release level earlier than V2R2M0, which is not supported by the SAVLICPGM command.

6E A RSTLICPGM command did not complete successfully. A preoperation exit program failed. The product being replaced had been installed.

6F A RSTLICPGM command failed. The failure was not a preoperation exit program or postoperation exit program. The product being replaced had been installed.

60 The product load (*PRDLOD) object for this load was loaded onto the system by the RSTLICPGM command.

61 The product load (*PRDLOD) object for this load was loaded onto the system by the RSTLICPGM command, but a postoperation exit program failed.

62 An installed library or folder was renamed, but the product does not allow dynamic naming. (A product specifies whether or not it allows dynamic naming when the product definition object is created.)

63 The product load (*PRDLOD) object for this load was installed by the RSTLICPGM command, but an object is damaged.

64 The product load (*PRDLOD) object for this load was installed by the RSTLICPGM command, but one of the following occurred:

- An object was found to be missing when the CHKPRDOPT command or the SAVLICPGM command was used.
- An error occurred while the DLTLICPGM command was being used.

67 The CHKPRDOPT command was used for this product load, but the postoperation exit program failed or indicated that an error was found.

90 The product load was installed successfully. If an object was missing or was damaged, and the problem was corrected, using the CHKPRDOPT command sets the state back to 90.

Load type. The type of load for which information was returned.

The possible values are:

**CODE* The load is a code load.

**LNG* The load is a language load.

Message file library name. The name of the library for the message file that contains the messages describing the product and its options.

Message file object name. The name of the message file that contains the messages describing the product and its options.

Minimum target release. The minimum release of the operating system to which the Save Licensed Program (SAVLICPGM) command will allow the product to be saved. The format must be in the format V*x*R*y*M*z*. Valid values for *x* and *y* are 0 through 9. Valid values for *z* are 0 through 9 and A through Z.

Minimum required VRM of option. The minimum release level that is allowed for the option that will run with the current level of the *BASE option for the product. This field is only applicable if mixed releases are allowed.

The possible values are:

**MATCH* The release of the option matches that of the *BASE.

VxRyMz The release value is in the format V*x*R*y*M*z*.

Minimum VRM of *BASE required by option. The minimum release level that is allowed for the *BASE option that will run with the current level of the option for the product. This field is only applicable if mixed releases are allowed and for a load type of *CODE.

The possible values are:

**MATCH* The release of the option matches that of the *BASE.

VxRyMz The release value is in the format V*x*R*y*M*z*.

Number of load records. The number of loads for this product option. The receiver variable may not have been large enough to hold all the load records. If this happens, this number may be larger than the number of load records actually returned.

Number of objects. The number of packaged objects for this load. The receiver variable may not have been

large enough to hold all the objects. If this happens, this number may be larger than the number of objects actually returned.

Number of option records. The number of options for this product and release level. The receiver variable may not have been large enough to hold all the option records. If this happens, this number may be larger than the number of option records actually returned.

Number of preoperation exit program names. The number of preoperation exit programs for this load for this primary library. If there are no preoperation exit programs for this library, this will be 0.

Number of primary folders. The number of primary folders for this load. The receiver variable may not have been large enough to hold all the folder records. If this happens, this number may be larger than the number of folder records actually returned.

Number of primary full paths. The number of full paths for this load. The receiver variable may not have been large enough to hold all the directory information. If this happens, this number may be larger than the number of directories actually returned.

Number of primary libraries. The number of primary libraries for this load. The receiver variable may not have been large enough to hold all the library information. If this happens, this number may be larger than the number of libraries actually returned. The first record contains the principal primary library information. Subsequent records contain the information for the additional libraries, if the load has any additional libraries.

Number of public object authorities available. The number of public object authorities associated with a product directory. This will be set to 0 if the product load has not been successfully packaged.

Number of releases returned. The number of release levels returned for format PRDR0700.

Object name. The name of an object for this load.

Object records. The objects in the object record are ordered by library. All objects for the principal library are first. Within each library, the objects are ordered by object type, but with product loads first and product definitions second, followed by all other object types.

The record has the following fields:

- Object name
- Installed library name
- Object type
- Reserved

Object type. The symbolic object type of the object.

Offset to additional information. The offset from the beginning of the receiver variable to the start of the rest of the information for a given format. This is to allow for expansion of the basic information. For format PRDR0100, this is 0.

Offset to directory information array. The offset from the beginning of the receiver variable to the start of the directory information array for format PRDR0800. This is to allow for expansion of the basic directory information.

Offset to folder records. The offset from the beginning of the receiver variable to the start of the first folder record for format PRDR0300. This is to allow for expansion of the basic folder information.

Offset to installed full path name. The offset from the beginning of the receiver variable to the start of the installed full path name for format PRDR0800. This is to allow for expansion of the basic directory information.

Offset to installed product directory name. The offset from the beginning of the receiver variable to the start of the installed product directory name for format PRDR0800. This will be the product directory path that follows the delimiter at the end of the product home directory. This offset will be equal to the length of the installed full path plus the offset to the installed full path if there is no product directory. This is to allow for expansion of the basic directory information.

Offset to library records. The offset from the beginning of the receiver variable to the start of the first library record for format PRDR0200. This is to allow for expansion of the basic library information.

Offset to load records. The offset from the beginning of the receiver variable to the start of the first load record for format PRDR0600. This is to allow for expansion of the basic load information.

Offset to next library record. The offset from the beginning of the receiver variable to the start of the next library record for format PRDR0200. If there are no more library records, then this is 0.

Offset to object records. The offset from the beginning of the receiver variable to the start of the first object record for format PRDR0400. This is to allow for expansion of the basic object information.

Offset to option records. The offset from the beginning of the receiver variable to the start of the first option record for format PRDR0500. This is to allow for expansion of the basic option information.

Offset to primary full path name. The offset from the beginning of the receiver variable to the start of the primary full path name for format PRDR0800. This is to allow for expansion of the basic directory information.

Offset to primary product directory name. The offset from the beginning of the receiver variable to the start of the primary product directory name for format PRDR0800. This will be the product directory path that follows the delimiter at the end of the product home directory. This offset will be equal to the length of the primary full path plus the offset to the primary full path if there is no product directory. This is to allow for expansion of the basic directory information.

Offset to public object authority array. The offset from the beginning of the receiver variable to the start of the public object authority array for format PRDR0800. This is to allow for expansion of the primary product home directory information.

Option records.
An array for which each entry includes the following fields:

- Product option
- Allow dynamic naming
- Product option message ID
- Minimum required VRM of option
- Reserved

Postoperation exit program name. The name of the postoperation exit program for this load for this primary library. If there is no postoperation exit program for this library, this will be blank.

Preoperation exit program names. An array of the preoperation exit programs for this load for this primary library. If there are no preoperation exit programs for this library, this will be an array of length 0.

Primary folder. The name of a primary folder for this load.

Primary full path name. The name of the primary full path. It contains a directory name in the CCSID that is indicated in the CCSID of returned directories field. The primary full path contains the primary product home directory concatenated with a slash (/), which is concatenated with the primary product directory name. The length of the product directory is the length of the primary full path plus the offset to the primary full path minus the offset to the primary product directory name.

Primary language load identifier. For code loads, this field contains the primary language of the product option. This is the National Language Version (NLV) of the language that is installed in the libraries. It will be blank if no language is installed in the libraries for the code load. For language loads (29xx), this field will always be blank.

Primary library name. The name of the primary library that was specified when the product load object was created.

Product ID. The product ID for which information was returned.

Product option. The product option for which information was returned.

Product option message ID. The message ID associated with this product option. The message ID was specified when the product definition was created.

Public data authority. The public data authority given to the directory by the Restore Licensed Program (RSTLICPGM) command when this load is installed if the directory does not exist. If the product load has not been successfully packaged, this field is blank.

Possible values follow:

*RWX	The directory is created with the public data authority set to *RWX.
*RW	The directory is created with the public data authority set to *RW.
*RX	The directory is created with the public data authority set to *RX.
*WX	The directory is created with the public data authority set to *WX.
*R	The directory is created with the public data authority set to *R.
*W	The directory is created with the public data authority set to *W.
*X	The directory is created with the public data authority set to *X.
*EXCLUDE	The directory is created with the public data authority set to *EXCLUDE.
*NONE	The directory is created with the public data authority set to *NONE.

Public object authorities array. The public object authority given to the directory by the Restore Licensed Program (RSTLICPGM) command when this load is installed if the directory does not exist. The number of elements is the number of public object authorities for this product directory. If the product load has not been successfully packaged, the number of public object authorities will be set to 0.

Possible values follow:

**NONE*	The directory is created with the public object authority set to *NONE.
**ALL*	The directory is created with the public object authority set to *ALL.
**OBJEXIST*	The directory is created with the public object authority set to *OBJEXIST.
**OBJMGT*	The directory is created with the public object authority set to *OBJMGT.
**OBJALTER*	The directory is created with the public object authority set to *OBJALTER.
**OBJREF*	The directory is created with the public object authority set to *OBJREF.

Registration type. The registration type associated with the product. The registration type and registration value together make up the registration ID for the product.

The possible values are:

02 Registration type *PHONE was specified when the product load or product definition was created.

04 The registration value is the same as the registration value for OS/400.

08 Registration type *CUSTOMER was specified when the product load or product definition was created.

Registration value. The registration value associated with the product. The registration type and registration value together make up the registration ID for the product.

Release date. Indicates the value specified for the release date when the product definition for this product load was created. The release date is in the format *yymmdd*, where *yy* equals year, *mm* equals month, and *dd* equals day. If no release date was specified when the product definition was created, then the release date is blank.

Release date century. The century that corresponds to the release date of the product.

Possible values follow:

0 Indicates years 19*xx*

1 Indicates years 20*xx*

Blank Indicates no release date was specified for the product.

Release level. The release level of the product for which information was returned. For V2R3M0, when format PRDR0700 is requested, the valid values for this field are *CUR, *PRV, V1R3M0, V2R1M0, V2R1M1, V2R2M0, and V2R3M0.

Release level (for each returned operating system release level). The individual release level returned. One or more may be returned.

Requirements met between base and option value. When a product allows mixed releases between its base and option, certain requirements must be met. This value represents the reason why the release requirements between the base and option may or may not be in error.

The possible values are:

0 There is not enough information available to determine if the release requirements have been met. This will be the value if this is a load type of *LANG.

1 The releases of the *BASE and option meet all requirements.

2 The release of the option is too old compared to the *BASE.

3 The release of the *BASE is too old compared to the option.

Reserved. An ignored field.

Secondary language library name. The secondary language library name that was specified when the load was created. If this is a code load, the secondary language library name is blank.

Supported flag. Whether this load is currently supported. A load can be supported by using the Work with Supported Products (WRKSPTPRD) command in the System Manager for iSeries licensed program.

The possible values are:

0 The load is not supported.

1 The load is supported.

Symbolic load state. The symbolic state of the load for which information was returned. This value, in conjunction with the load error indicator, can be used to determine if the load is installed correctly.

The possible values are:

**DEFINED* The load is defined. The product load object for this load does not exist. When a product definition is created, a code load is defined for each product option, and language loads can be defined.

**CREATED* The product load object for this load exists. It must be packaged with the PKGPRDOPT command before it can be saved using the SAVLICPGM command.

**PACKAGED* The product load object for this load has been packaged with the PKGPRDOPT command.

**DAMAGED* If this is for an option other than the base option or for a language load for the base option, the product load object has been damaged. If this is for the code load for the base option, one of the following happen:

- The product definition for this product ID and release level has been damaged.
- The product load object has been damaged.

**LOADED* Indicates one of the following:

- A restore licensed program function is in progress.
- A delete licensed program function is in progress.
- The product was created previous to V2R2M0, and there was an error during the process of converting the product information.

**INSTALLED* The product load (*PRDLOD) object for this load was loaded onto the system by the RSTLICPGM command.

Error Messages

Message ID	Error Message Text
CPF0C1B E	Requirements between parameters not satisfied.
CPF0C1C E	Release level &1 not valid.
CPF0C1D E	Load ID &1 not valid.
CPF0C1E E	Error occurred during running of &1 API.
CPF0C1F E	Product information not found.
CPF0C30 E	Release level *ONLY not valid for product &1.
CPF0C4B E	Product availability object &2/&1 recovery required.

CPF0C4C E	Cannot allocate object &1 in library &2.
CPF0C4D E	Error occurred while processing object &1 in library &2.
CPF247E E	CCSID &1 is not valid.
CPF24B4 E	Severe error while addressing parameter list.
CPF3CF1 E	Error code parameter not valid.
CPF3C1D E	Length specified in parameter &1 not valid.
CPF3C19 E	Error occurred with receiver variable specified.
CPF3C21 E	Format name &1 is not valid.
CPF3C24 E	Length of the receiver variable is not valid.
CPF3C39 E	Value for reserved field not valid.
CPF3C90 E	Literal value cannot be changed.
CPF8191 E	Product definition &4 in &9 damaged.
CPF8193 E	Product load object &4 in &9 damaged.
CPF9838 E	User profile storage limit exceeded.
CPF9872 E	Program or service program &1 in library &2 ended. Reason code &3.

API Introduced: V2R3

Top | Software Product APIs | APIs by category

Retrieve Program Temporary Fix Information (QPZRTVFX) API

Required Parameter Group:

1	Receiver variable	Output	Char(*)
2	Length of receiver variable	Input	Binary(4)
3	PTF information	Input	Char(50)
4	Format name	Input	Char(8)
5	Error Code	I/O	Char(*)

Default Public Authority: *USE

Threadsafe: No

The Retrieve Program Temporary Fix Information (QPZRTVFX) API returns information about a specific program temporary fix (PTF). The information returned is determined by the format specified.

You can use the QPZRTVFX API to:

- Retrieve basic information about a PTF.
- Retrieve the cover letter information for a PTF.
- Retrieve the requisites for a PTF.
- Retrieve the list of objects for a PTF.
- Retrieve the dependents for a PTF.
- Retrieve the list of APARs for a PTF.
- Retrieve the list of symptom strings for a PTF.
- Retrieve the list of exit programs for a PTF.
- Retrieve the preconditions for a PTF.
- »Retrieve the superseded PTF IDs for a PTF.«

Authorities and Locks

None.

Lock conflicts may occur if this API is called while another PTF operation is in progress.

Required Parameter Group

Receiver variable

OUTPUT; CHAR(*)

The receiver variable that is to receive the information requested. The length of this area must be passed in the length of receiver variable parameter. The API returns only the data the area can hold.

Length of receiver variable

INPUT; Binary(4)

The length of the receiver variable. You can specify a smaller area than the format requested as long as you specify the length of receiver variable parameter correctly. If the length specified is larger than the size of the receiver variable, the results are not predictable. This value must be greater than or equal to 8.

PTF information

INPUT; CHAR(50)

The attributes of the PTF for which information is being requested. For more information on this parameter see Format of PTF Information.

Format name

INPUT; CHAR(8)

The content and format of the information returned for the PTF. The possible format names are:

PTFR0100 Basic information; for details, see PTFR0100 Format.

PTFR0200 This format contains the basic information as well as an array of cover letters in the format given. If the receiver variable does not have enough space for all the cover letters to be returned, only the cover letters that fit in the space provided are returned. For details, see PTFR0200 Format.

PTFR0300 This contains the basic information plus an array that contains information about the requisite PTFs. If the receiver variable does not have enough space to return all requisite information, only the information that fits in the space provided is returned. For details, see PTFR0300 Format.

PTFR0400 This contains the basic information plus an array that contains a list of the PTF objects and their alternative names. If the receiver variable does not have enough space to return all of the PTF object information, only the information that fits in the space provided is returned. For details, see PTFR0400 Format.

PTFR0500 This contains the basic information plus an array that contains information about the dependent PTFs. If the receiver variable does not have enough space to return all dependent information, only the information that fits in the space provided is returned. For details, see PTFR0500 Format.

PTFR0600 This contains the basic information plus an array that contains information about the APAR numbers. If the receiver variable does not have enough space to return all APAR information, only the information that fits in the space provided is returned. For details, see PTFR0600 Format.

PTFR0700 This contains the basic information plus an array that contains information about the symptom strings. If the receiver variable does not have enough space to return all symptom string information, only the information that fits in the space provided is returned. For details, see PTFR0700 Format.

PTFR0800 This contains the basic information plus an array that contains the exit program information. If the receiver variable does not have enough space to return all exit program information, only the information that fits in the space provided is returned. For details, see PTFR0800 Format.

PTFR0900 This contains the basic information plus an array that contains the information about the preconditions. If the receiver variable does not have enough space to return the precondition information, only the information that fits in the space provided is returned. For details, see PTFR0900 Format.

»*PTFR1000* This contains the basic information plus an array that contains the information about the superseded PTFs. If the receiver variable does not have enough space to return the superseded PTF information, only the information that fits in the space provided is returned. For details, see PTFR1000 Format.«

Error code

I/O; CHAR(*)

The structure in which to return error information. For the format of the structure, see Error Code Parameter.

Format of PTF Information

For detailed descriptions of each field, see Field Descriptions.

Offset		Type	Field
Dec	Hex		
0	0	CHAR(7)	PTF ID
7	7	CHAR(7)	Product ID
14	E	CHAR(6)	Release level
20	14	BINARY(4)	CCSID for returned directory names
24	18	CHAR(1)	Close PTF database files
25	19	CHAR(25)	Reserved

Field Descriptions

CCSID for returned directory names. The CCSID in which the directory names should be returned. If this field is blank, the CCSID is the job default CCSID. This value is used only for returned directory names.

Close PTF database files. Whether the PTF database files should remain open after returning from the API call. This field allows for improved performance when this API is called numerous times during processing. Do not leave the PTF database files open for long periods of time because other PTF operations cannot be performed while these files are open. If a blank is specified in this field, it will be treated as a 0.

0 The PTF database files will be closed before returning from the API call.

1 The PTF database files will remain open until the API is called again with this field set to 0.

Product ID. The product ID for the PTF for which information is requested. The possible values are:

**ONLY* The product ID is not known, but only one PTF exists on the system by this PTF ID.

product ID The product ID for the PTF.

PTF ID. The identifier of the PTF for which information is requested.

Release level. The release of the PTF for which information is requested. This field is ignored if *ONLY is specified in the product ID field. The format is:

VxRyMz The release of the PTF is in the format VxRyMz. Valid entries for x and y are any number between 0 and 9. A valid entry for z is a number between 0 and 9 or a character between A and Z.

Reserved. This field must contain blanks.

PTFR0100 Format

This format returns basic information about the PTF. For detailed descriptions of each field, see Field Descriptions.

Offset		Type	Field
Dec	Hex		
0	0	BINARY(4)	Bytes returned
4	4	BINARY(4)	Bytes available
8	8	BINARY(4)	Offset to additional information
12	C	CHAR(7)	Product ID
19	13	CHAR(7)	PTF ID
26	1A	CHAR(6)	Release level
32	20	CHAR(4)	Product option
36	24	CHAR(4)	Load ID
40	28	CHAR(1)	Loaded status
41	29	CHAR(1)	Cover letter status
42	2A	CHAR(1)	On-order status

43	2B	CHAR(1)	Save file status
44	2C	CHAR(10)	File name
54	36	CHAR(10)	File library name
64	40	CHAR(1)	PTF type
65	41	CHAR(1)	IPL action
66	42	CHAR(1)	Action pending
67	43	CHAR(1)	Action required
68	44	CHAR(1)	PTF is released
69	45	CHAR(6)	OS/400 target release
75	4B	CHAR(7)	Superseding PTF
82	52	CHAR(1)	Current IPL source
83	53	CHAR(2)	Minimum level
85	55	CHAR(2)	Maximum level
87	57	CHAR(1)	Format information available
88	58	CHAR(13)	Status date and time
101	65	CHAR(7)	Licensed Internal Code group
»108	6C	CHAR(7)	Superseded by PTF ID «
»115	73«	CHAR(*)	Reserved

PTFR0200 Format

The fields that follow the cover letter records field define an entry in that array. That group of fields is repeated by the number of different NLVs available for the cover letter field. For detailed descriptions of each field, see Field Descriptions.

Offset		Type	Field
Dec	**Hex**		
0	0		All information from format PTFR0100
Note: The decimal and hexadecimal offsets are reached by using the offset to additional information field in format PTFR0100.			
		BINARY(4)	Offset to first cover letter record from the beginning
		BINARY(4)	Number of different NLVs available for the cover letter
		BINARY(4)	Length of cover letter record
		CHAR(*)	Reserved
Note: The decimal and hexadecimal offsets are determined by using the value in the offset to first cover letter of format PTFR0200.			
		ARRAY of CHAR(*)	Cover letter records
		CHAR(4)	Cover letter national language version (NLV)
		CHAR(10)	Cover letter file name

	CHAR(10)	Cover letter library name
	CHAR(10)	Cover letter member name
	CHAR(1)	Pre-apply or pre-remove considerations
	CHAR(1)	Post-apply or post-remove considerations
	CHAR(*)	Reserved

PTFR0300 Format

The fields following the requisite record field define an entry in that array. That group of fields is repeated by the number of requisites. For detailed descriptions of each field, seeField Descriptions.

Offset		Type	Field
Dec	**Hex**		
0	0		All information from format PTFR0100
Note: The decimal and hexadecimal offsets are reached by using the offset to additional information field in format PTFR0100.			
		BINARY(4)	Offset to first requisite record
		BINARY(4)	Number of requisites
		BINARY(4)	Length of requisite record
		CHAR(*)	Reserved
Note: The decimal and hexadecimal offsets are determined by using the value in the offset to first requisite record of format PTFR0300.			
		ARRAY of CHAR(*)	Requisite record
		CHAR(7)	Requisite product ID
		CHAR(7)	Requisite PTF ID
		CHAR(6)	Release of requisite
		CHAR(2)	Requisite minimum level
		CHAR(2)	Requisite maximum level
		CHAR(1)	Type of requisite
		CHAR(1)	Requisite is conditional
		CHAR(1)	Requisite is required
		CHAR(4)	Requisite option
		CHAR(4)	Requisite load ID
		CHAR(*)	Reserved

PTFR0400 Format

The fields following the PTF object record field define an entry in that array. That group of fields is repeated by the number of PTF objects. If a PTF contains subobjects, an offset to an array of subobjects is provided. The subobject fields are repeated by the number of PTF subobjects. For detailed descriptions of each field, see Field Descriptions.

Offset		Type	Field
Dec	Hex		
0	0		All information from format PTFR0100
Note: The decimal and hexadecimal offsets are reached by using the offset to additional information field in format PTFR0100.			
		BINARY(4)	Offset to the first PTF object record
		BINARY(4)	Number of PTF objects
		BINARY(4)	Length of PTF object record
		BINARY(4)	CCSID of returned directories
		CHAR(1)	Error indicator for CCSID conversion
		CHAR(*)	Reserved
Note: The decimal and hexadecimal offsets are determined by using the value in the offset to the first PTF object record of format PTFR0400.			
		ARRAY of CHAR(*)	PTF object record
		BINARY(4)	Offset to this PTF object's first subobject record
		BINARY(4)	Number of PTF subobjects
		BINARY(4)	Length of subobject record
		CHAR(10)	Object name
		CHAR(10)	Object library
		CHAR(10)	Alternative object name
		CHAR(7)	Object type
		CHAR(*)	Reserved
Note: The decimal and hexadecimal offsets are determined by using the value in the offset to this PTF object's first subobject record of format PTFR0400.			
		ARRAY of CHAR(*)	PTF subobject record
		BINARY(4)	Offset to subobject name
		BINARY(4)	Length of subobject name
		CHAR(7)	Subobject attribute
		CHAR(*)	Reserved
Note: The decimal and hexadecimal offsets are determined by using the value in the offset to the subobject name in the subobject array record of format PTFR0400.			
		CHAR(*)	Subobject name

PTFR0500 Format

The fields following the dependent record field define an entry in that array. That group of fields is repeated by the number of dependents. For detailed descriptions of each field, see Field Descriptions.

Offset		Type	Field
Dec	Hex		
0	0		All information from format PTFR0100
Note: The decimal and hexadecimal offsets are reached by using the offset to additional information field in format PTFR0100.			
		BINARY(4)	Offset to first dependent PTF record
		BINARY(4)	Number of dependent PTFs
		BINARY(4)	Length of dependent PTF record
		CHAR(*)	Reserved
Note: The decimal and hexadecimal offsets are determined by using the value in the offset to first dependent PTF record of format PTFR0500.			
		ARRAY of CHAR(*)	Dependent PTF record
		CHAR(7)	Dependent product ID
		CHAR(7)	Dependent PTF ID
		CHAR(6)	Release of dependent PTF
		CHAR(2)	Dependent minimum level
		CHAR(2)	Dependent maximum level
		CHAR(1)	Type of dependent PTF
		CHAR(4)	Dependent option
		CHAR(4)	Dependent load ID
		CHAR(*)	Reserved

PTFR0600 Format

The fields following the APAR record field define an entry in that array. That group of fields is repeated by the number of APAR records. For detailed descriptions of each field, see Field Descriptions.

Offset		Type	Field
Dec	Hex		
0	0		All information from format PTFR0100
Note: The decimal and hexadecimal offsets are reached by using the offset to additional information field in format PTFR0100.			
		BINARY(4)	Offset to first APAR record
		BINARY(4)	Number of APAR records
		BINARY(4)	Length of APAR record
		CHAR(*)	Reserved

Note: The decimal and hexadecimal offsets are determined by using the value in the offset to the first APAR record of format PTFR0600.		
	ARRAY of CHAR(*)	APAR record
	CHAR(7)	APAR number
	CHAR(*)	Reserved

PTFR0700 Format

The fields following the symptom string record field define an entry in that array. That group of fields is repeated by the number of symptom string records. For detailed descriptions of each field, see Field Descriptions.

Offset		Type	Field
Dec	**Hex**		
0	0		All information from format PTFR0100
Note: The decimal and hexadecimal offsets are reached by using the offset to additional information field in format PTFR0100.			
		BINARY(4)	Offset to first symptom string record
		BINARY(4)	Number of symptom string records
		BINARY(4)	Length of symptom string record
		CHAR(*)	Reserved
Note: The decimal and hexadecimal offsets are determined by using the value in the offset to the first symptom string record of format PTFR0700.			
		ARRAY of CHAR(*)	Symptom string record
		BINARY(4)	Offset to symptom string data
		BINARY(4)	Length of symptom string data
		CHAR(*)	Reserved
Note: The decimal and hexadecimal offsets are determined by using the value in the offset to the symptom string data of format PTFR0700.			
		CHAR(*)	Symptom string data

PTFR0800 Format

The fields following the exit program record field define an entry in that array. That group of fields is repeated by the number of exit programs. For detailed descriptions of each field, see Field Descriptions.

Offset		Type	Field
Dec	**Hex**		
0	0		All information from format PTFR0100

Note: The decimal and hexadecimal offsets are reached by using the offset to additional information field in format PTFR0100.		
	BINARY(4)	Offset to first exit program record
	BINARY(4)	Number of exit program records
	BINARY(4)	Length of exit program record
	CHAR(*)	Reserved
Note: The decimal and hexadecimal offsets are determined by using the value in the offset to the first exit program record of format PTFR0800.		
	ARRAY of CHAR(*)	Exit program record
	BINARY(4)	Offset to exit program user data
	BINARY(4)	Length of exit program user data
	CHAR(10)	Exit program name
	CHAR(10)	Exit program library
	CHAR(1)	Exit program run option
	CHAR(*)	Reserved
Note: The decimal and hexadecimal offsets are determined by using the value in the offset to the exit program user data of format PTFR0800.		
		CHAR(*)	Exit program user data

PTFR0900 Format

The fields following the precondition record field define an entry in that array. That group of fields is repeated by the number of precondition records. For detailed descriptions of each field, see Field Descriptions.

Offset		Type	Field
Dec	**Hex**		
0	0		All information from format PTFR0100
Note: The decimal and hexadecimal offsets are reached by using the offset to additional information field in format PTFR0100.			
		BINARY(4)	Offset to first precondition record
		BINARY(4)	Number of precondition records
		BINARY(4)	Length of precondition record
		CHAR(*)	Reserved
Note: The decimal and hexadecimal offsets are determined by using the value in the offset to the first precondition record of format PTFR0900.			
		ARRAY of CHAR(*)	Precondition record
		CHAR(10)	Precondition type
		CHAR(10)	Precondition name
		CHAR(10)	Precondition library name
		CHAR(*)	Reserved

»PTFR1000 Format

The fields following the superseded PTF record field define an entry in that array. That group of fields is repeated by the number of superseded PTF records. For detailed descriptions of each field, see Field Descriptions.

Offset		Type	Field
Dec	**Hex**		
0	0		All information from format PTFR0100
Note: The decimal and hexadecimal offsets are reached by using the offset to additional information field in format PTFR0100.			
		BINARY(4)	Offset to first superseded PTF record
		BINARY(4)	Number of superseded PTF records
		BINARY(4)	Length of superseded PTF record
		CHAR(*)	Reserved
Note: The decimal and hexadecimal offsets are determined by using the value in the offset to the first superseded PTF record of format PTFR1000.			
		ARRAY of CHAR(*)	Superseded PTF record
		CHAR(7)	Superseded PTF ID
		CHAR(*)	Reserved

«

Field Descriptions

APAR number. The number of the APAR that is fixed when you install this PTF.

APAR record. The information about the APAR. The record is an array that contains the fields in format PTFR0600.

Action pending. Whether a required action has yet to be performed to make this PTF active. This field reflects the current status of any required actions. The following values are valid:

0 No required actions are pending for this PTF.

1 A required action needs to occur for this PTF to be active.

Check the activation instructions section of the cover letter to determine what the action is.

If the action required field is a 2 and you have performed the activation instructions, then the PTF is active. However, this field will not be updated until the next IPL.

Action required. Whether an action is required to make this PTF active when it is applied. See the cover letter to determine what action needs to be taken. The following values are valid.

0 No activations instructions are needed for this PTF.

1 This PTF was shipped with activation instructions in the cover letter. This PTF has an exit program to update the status of the PTF after the activation instructions have been performed.

2 This PTF was shipped with activation instructions in the cover letter. No exit program exists to verify the activation instructions were performed.

Alternative object name. The alternative name of the object when the PTF is temporarily applied or temporarily removed. PTFs that have been temporarily applied have alternative object names in the format of QPZA*nnnnnn*; PTFs that are temporarily removed have alternative object names in the format of QPZR*nnnnnn*, where *nnnnnn* is some number. This field is blank for the following reasons:

- the PTF is not temporarily applied or is temporarily removed.
- the object is a temporary object on the system; that is, an object name that starts with QPZ1 or QPZ2.
- the object was bypassed when the PTF was loaded.

Bytes available. The number of bytes of data available to be returned to the user.

Bytes returned. The number of bytes that were returned to the user. This is the lesser of the number of bytes available to be returned or the length of the receiver variable.

CCSID of returned directories. The value of the CCSID in which the directories were returned. This will be the requested CCSID if the value of the CCSID conversion indicator is 0. This field is used only for subobject attributes of DIR (directory) and SOM (system object model).

Cover letter file name. The file containing the cover letter member.

Cover letter library name. The library containing the cover letter file.

Cover letter member name. The member containing the cover letter.

Cover letter national language version (NLV). The NLV corresponds to a code that indicates the language in which the cover letter was written. For more information on NLVs, see the Globalization topic in the iSeries Information Center.

Cover letter records. The record containing the cover letter information.

Cover letter status. Whether a cover letter exists for the PTF. The following values are valid.

0 The PTF has no cover letter.

1 The PTF has a cover letter.

Current IPL source. The copy of Licensed Internal Code that the system is currently operating from. The previous IPL of the system used this copy of Licensed Internal Code.

A The system is currently operating on the A IPL source.

B The system is currently operating on the B IPL source.

Blank The current IPL side could not be determined. A blank will be returned, not the word blank.

Dependent load ID. The load ID of the dependent PTF. This value is blank when the load ID of the dependent PTF cannot be determined.

Dependent maximum level. The indicator of the highest level of the product for which this PTF can be

installed. This field will be blank if the product has no level.

Dependent minimum level. The indicator of the lowest level of the product for which this PTF can be installed. This field will be blank if the product has no level.

Dependent option. The product option of the dependent PTF. This value is blank when the option of the dependent PTF cannot be determined.

Dependent product ID. The product ID of the dependent PTF.

Dependent PTF ID. The PTF ID of the dependent PTF.

Dependent PTF record. The information about the dependent PTF. The record is an array that contains the fields in format PTFR0500.

Error indicator about CCSID conversion. Whether the CCSID conversion to the requested CCSID was successful. If the requested CCSID conversion fails, the CCSID in which the directories are returned is identified in the CCSID of returned directories field. This field is used only for subobject attributes of DIR (directory) and SOM (system object model). Possible values follow:

0 CCSID conversion was successful.

1 CCSID conversion failed.

Exit program library. The name of the library containing the exit program.

Exit program name. The name of a program that will be run at certain stages of the PTF process.

Exit program record. The information about the exit program. The record is an array that contains the fields in format PTFR0800.

Exit program run option.

The stage of the PTF process in which the exit program will be run. Possible values follow:

0 *ACTION - The exit program is called to determine if there is action necessary to make the PTF active or inactive.

1 *BOTH - The exit program will be run at the end of apply and remove processing.

2 *APPLY - The exit program will be run at the end of apply processing.

3 *REMOVE - The exit program will be run at the end of remove processing.

4 *PREAPY - The exit program will be run before the PTF is applied and at the end of apply processing.

5 *PRERMV - The exit program will be run before the PTF is removed and at the end of remove processing.

6 *PREBTH - The exit program will be run before the PTF is removed and at the end of remove processing. It is also run before the PTF is applied and at the end of apply processing.

File library name. The name of the library where the save file for the PTF is located. If no save file name has been reserved, this field will be blank.

File name. The name of the file where the save file for the PTF is located. If no save file name has been

reserved, this field will be blank.

Format information available. When you request information using formats PTFR0300, PTFR0400, PTFR0600, PTFR0700, PTFR0800, PTFR0900,» or PTFR1000« this field indicates whether the system has the information available to return for the PTF. The system may not be able to return the requested information if the PTF is permanently applied or superseded and does not have a save file. The possible values are:

0 There was not enough information available about this PTF to return the requested data.

1 The information is available and complete. Refer to specific format fields for the details.

IPL action. The action to be taken on this PTF during the next IPL. The following values are valid:

0 No action will occur at the next IPL.

1 The PTF will be temporarily applied at the next IPL.

2 The PTF will be temporarily removed at the next IPL.

3 The PTF will be permanently applied at the next IPL.

4 The PTF will be permanently removed at the next IPL.

Length of APAR record. The length of each APAR record.

Length of cover letter record. The length of each cover letter record.

Length of dependent PTF record. The length of each dependent PTF record.

Length of exit program record. The length of each exit program record.

Length of precondition record. The length of each precondition record.

Length of PTF object record. The length of one array element in the PTF object record.

Length of requisite record. The length of each requisite record.

Length of subobject name. The length of the subobject name.

Length of subobject record. The length of one array element in the subobject record.

»**Length of superseded PTF record.** The length of each superseded PTF record.«

Length of symptom string data. The length of the returned symptom string.

Length of symptom string record. The length of each symptom string record.

Length of exit program user data. The length of the returned exit program user data string.

Licensed Internal Code Group. The name of the Licensed Internal Code Group for this PTF. If the name of the group is not available or if the PTF is not a Licensed Internal Code fix, this field will be blank.

Loaded status. The current loaded status of the PTF. A PTF can have any of the following statuses:

0 The PTF has never been loaded.

1 The PTF has been loaded.

2 The PTF has been applied.

3 The PTF has been applied permanently.

4 The PTF has been permanently removed.

5 The PTF is damaged. An error occurred while applying the PTF. It needs to be reloaded and applied.

6 » The PTF is superseded. A PTF will have a status of superseded when one of the following situations occurs :

- Another PTF with a more recent correction for the problem has been loaded on the system. The identifier of the PTF that has been loaded can be found in the superseded by PTF ID field.
- The PTF save file for another PTF with a more recent correction for the problem has been logged into *SERVICE on the system. The most recent PTF ID with a PTF save file that has been logged into *SERVICE can be found in the superseding PTF field.

«

Note: These fields are returned as numbers instead of text because statuses are translatable text instead of special values. The text message that contains these values is CPX3501.

This field may be blank.

Load ID. The load ID of the product load for the PTF.

Maximum level. The indicator of the highest level of the product on which this PTF can be installed. If the minimum and maximum levels are the same, then this PTF can only be installed on one level of the product. The level can be AA to 99. This field will be blank if the product has no level.

Minimum level. The indicator of the lowest level of the product on which this PTF can be installed. If the minimum and maximum levels are the same, then this PTF can only be installed on one level of the product. The level can be AA to 99. This field will be blank if the product has no level.

Number of APAR records. The number of returned APAR records.

Check the format information available field returned in PTFR0100 before using this value. The number of APARs will be zero when the system cannot determine if this PTF has APARs.

Number of dependent PTFs. The number of dependent PTFs available. The receiver variable may not have been large enough to hold all the dependent PTF records. If so, the number of dependents returned may be less than this value.

Number of different NLVs available for the cover letter. Cover letter member names are returned in an array. This number indicates how many NLVs were available for the cover letter. However, the length of the receiver may not have been large enough to hold all the cover letter NLVs available. If so, the number of cover letter member names returned may be less than this value.

Number of exit program records. The number of exit program records. The receiver variable may not have been large enough to hold all the exit program records. If so, the number of exit programs returned may be less than this value.

Check the format information available field returned in PTFR0100 before using this value. The number of exit programs will be zero when the system cannot determine if this PTF has any exit programs.

Number of precondition records. The number of returned precondition records.

Check the format information available field returned in PTFR0100 before using this value. The number of

precondition records will be zero when the system cannot determine if this PTF has preconditions.

Number of PTF objects. The number of PTF objects available. The receiver variable may not have been large enough to hold all the PTF object records. If so, the number of PTF objects returned may be less than this value.

Check the format information available field returned in PTFR0100 before using this value. The number of objects will be zero when the system cannot determine the number of objects contained in this PTF.

Number of PTF subobjects. The number of PTF subobjects associated with this PTF object. The receiver variable may not have been large enough to hold all the PTF subobject records. If so, the number of PTF subobjects returned may be less than this value.

Number of requisites. The number of requisite PTFs available. The receiver variable may not have been large enough to hold all the requisite PTF records. If so, the number of requisites returned may be less than this value.

Check the format information available field returned in PTFR0100 before using this value. The number of requisites will be zero when the system cannot determine if this PTF has requisite PTFs.

»**Number of superseded PTF records.** The number of superseded PTF records. The receiver variable may not have been large enough to hold all the superseded PTF records. If so, the number of superseded PTFs returned may be less than this value.

Check the format information available field returned in PTFR0100 before using this value. The number of superseded PTF records will be zero when the system cannot determine if this PTF has any superseded PTFs.«

Number of symptom string records. The number of returned symptom string records.

Check the format information available field returned in PTFR0100 before using this value. The number of symptom strings will be zero when the system cannot determine if this PTF has symptom strings.

Object library. The primary library where PTF objects will be placed for this product. If the product was installed in the primary library, this is the library where the PTF objects will reside.

Note: If the product was installed in a library other than the primary library, the API user might need to determine the installed library that corresponds to this primary library by using format PRDR0200 of the Retrieve Product Information (QSZRTVPR) API.

Object name. The name of the object that is contained in the PTF.

Object type. The symbolic type of the object that is contained in the PTF.

Offset to additional information. The offset from the beginning of the receiver variable to the start of either the cover letter information or the requisite information. This is to allow expansion of the basic information.

Offset to exit program user data. The offset from the beginning of the receiver variable to the start of the exit program user data.

Offset to first APAR record. The offset from the beginning of the receiver variable to the start of the APAR record.

Offset to first cover letter record from the beginning. The offset from the beginning of the receiver variable to the first cover letter record.

Offset to first dependent PTF record. The offset from the beginning of the receiver variable to the start of the first dependent PTF record.

Offset to first exit program record The offset from the beginning of the receiver variable to the start of the

first exit program record.

Offset to first precondition record. The offset from the beginning of the receiver variable to the first precondition record.

Offset to first PTF object record. The offset from the beginning of the receiver variable to the start of the PTF object record.

Offset to first requisite record. The offset from the beginning of the receiver variable to the start of the first requisite record.

»**Offset to first superseded PTF record** The offset from the beginning of the receiver variable to the start of the first superseded PTF record.«

Offset to first symptom string record. The offset from the beginning of the receiver variable to the start of the first symptom string record.

Offset to subobject name. The offset from the beginning of the receiver variable to the start of the subobject name.

Offset to symptom string data. The offset from the beginning of the receiver variable to the start of the symptom string data.

Offset to this PTF object's first subobject record. The offset from the beginning of the receiver variable to the start of the first PTF subobject record.

On-order status. Whether the PTF has been ordered. The following values are valid:

0 The PTF has not been ordered or has already been received.

1 The PTF has been ordered.

OS/400 target release. The earliest release of the operating system on which you can load and apply the PTF. The release level is specified in the format V*x*R*y*M*z*, where V*x* is the version, R*y* is the release, and M*z* is the modification level.

Post-apply or post-remove considerations. Whether the cover letter contains special instructions that should be followed after applying or removing the PTF.

0 The PTF cover letter does not have any post-apply or post-remove considerations.

1 The PTF cover letter does have post-apply or post-remove considerations.

9 It is not known if the PTF cover letter has any post-apply or post-remove considerations. The most likely reasons are that the PTF cover letter was created prior to operating system release V5R1M0, or the cover letter was created using the System Manager licensed product.

Pre-apply or pre-remove considerations. Whether the cover letter contains special instructions that should be followed prior to applying or removing the PTF.

0 The PTF cover letter does not have any pre-apply or pre-remove considerations.

1 The PTF cover letter does have pre-apply or pre-remove considerations that should be followed regardless of how the PTF is applied or removed (either immediately or during an IPL).

2 The PTF cover letter does have pre-apply or pre-remove considerations, but only when the PTF is applied or removed immediately.

3 The PTF cover letter does have pre-apply or pre-remove considerations, but only when the PTF is applied or removed during an IPL.

9 It is not known if the PTF cover letter has any pre-apply or pre-remove considerations. The most likely reasons are that the PTF cover letter was created prior to operating system release V5R1M0, or the cover letter was created using the System Manager licensed product.

Precondition library name. The name of the library for the precondition object. This field contains blanks when the precondition type is *JOB, *SBS, or *RSTD.

Precondition name. The name of the object, job, or subsystem for which the precondition exists. The name can be either a specific name or a generic name. A generic name is a character string that contains one or more characters followed by an asterisk(*). The field contains blanks when the precondition type is *RSTD.

Precondition record. The information about the precondition. Precondition records identify objects, jobs, or subsystems that cannot be active when this PTF is temporarily applied or removed immediately. The record is an array that contains the fields in format PTFR0900.

Precondition type. The type of the object indicated in the precondition name. Possible values are:

Object type	The precondition name specifies the name of an object of this type that exists in the precondition library.
**JOB*	The precondition name specifies the name of a job.
**SBS*	The precondition name specifies the name of a subsystem.
**RSTD*	The system must be in a restricted state to temporarily apply or temporarily remove this PTF immediately.

Product ID. The PTF is for this product. The product must be installed or supported.

Product option. The PTF is for this option of the product.

PTF ID. The identifier of the PTF.

PTF is released. Whether the PTF save file is available for distribution to another system. This is set to 1 only when the System Manager licensed product is on the system and the product is supported. The user needs to check the PTF save file status before using this field. Possible values are:

0 The PTF save file cannot be distributed.

1 The PTF save file is released and can be distributed to another system.

PTF object record. The information about the PTF objects. The record is an array that contains the fields described in format PTFR0400.

This information is available only if the PTF is loaded, temporarily applied, or has a save file.

PTF subobject record. The information about the PTF subobjects. The record is an array that contains the fields described in format PTFR0400.

PTF type. The type of PTF. The possible values are:

0 PTF is delayed. The PTF must be applied at IPL time.

1 The PTF is immediate. The PTF can be applied immediately. No IPL is needed.

Blank The PTF type is not known.

Release level. The release of the PTF. It must be in the format of VxRyMz. Valid entries for x and y are any number between 0 and 9. Valid entries for z is any number between 0 and 9 or a character between A and Z.

Release of dependent PTF. The release of the dependent PTF.

Release of requisite. The release of the requisite PTF.

Requisite is conditional. Whether the requisite relationship is conditional. Users should check this field to determine whether it is necessary to check a remote system for the presence of the software that is described in the product ID, release, option, and load ID.

0 The requisite relationship is not conditional. The requisite PTF is required by this PTF on all systems that can use this PTF.

1 The requisite relationship is conditional. The requisite PTF is required by this PTF only on systems that contain the software described in the other fields.

Requisite is required. Whether a requisite PTF is required on this system. The possible values are:

0 The requisite PTF is not required with this PTF on this system.

1 The requisite PTF is required with this PTF on this system.

Requisite load ID. The load ID of the requisite PTF. This value may be blank when the load ID of the requisite PTF cannot be determined.

Requisite option. The product option of the requisite PTF. This value may be blank when the option of the requisite cannot be determined.

Requisite maximum level. The indicator of the highest level of the product on which this PTF can be installed. This field will be blank if the product has no level.

Requisite minimum level. The indicator of the lowest level of the product on which this PTF can be installed. This field will be blank if the product has no level.

Requisite product ID. The product ID of the requisite PTF.

Requisite PTF ID. The PTF ID of the requisite PTF.

Requisite record. The information about the requisite. The record is an array that contains the fields in format PTFR0300.

Reserved. This field is ignored.

Save file status. Whether a save file exists for the PTF. This field should always be checked to determine if a save file exists. The following values are valid:

0 The PTF has no save file.

1 The PTF has a save file.

Status date and time. The date and time the PTF status was last changed. This field will be blank when the

status date and time is not available. The date and time field is in the CYYMMDDHHMMSS format:

C Century, where 0 indicates years 19*xx* and 1 indicates years 20*xx*.

YY Year

MM Month

DD Day

HH Hour

MM Minute

SS Second

Subobject attribute. The type of subobject contained in the PTF. The possible values follow:

DIR The PTF subobject is a directory.

DOC The PTF subobject is a document.

Subobject name. The name of the subobject. This is a name of an object that is not limited to 10 characters.

»**Superseded by PTF ID.** The identifier of the PTF that has replaced this PTF. This field will be blank when the PTF is not superseded or when the superseding PTF has not been loaded on the system.«

»**Superseded PTF ID.** The identifier of a PTF that is superseded by this PTF.«

»**Superseded PTF record.** The information about the superseded PTFs. The record is an array that contains the fields in format PTFR1000.«

Superseding PTF. » The identifier of the most recent supersede of this PTF that exists on the system. This field will be blank when the PTF does not have a superseding PTF.«

Symptom string data. The symptom string for the problems fixed by this PTF.

Symptom string record. The information about the symptom string. The record is an array that contains the fields in format PTFR0700.

Type of dependent PTF. The type of dependent relationship. The possible values are:

1 The PTF you are retrieving is a prerequisite for this PTF.

2 The PTF you are retrieving is a corequisite for this PTF.

Type of requisite. The type of requisite relationship. The possible values are:

1 The requisite PTF is a prerequisite of this PTF. It does not require this PTF.

2 The requisite PTF is a corequisite of this PTF. It does require this PTF.

»*9* The requisite PTF is a distribution requisite of this PTF. It requires this PTF for distribution purposes only.«

Error Messages

Message ID	Error Message Text
CPF0CB3 E	Value for reserved field not valid.
CPF24B4 E	Severe error while addressing parameter list.
CPF3CF1 E	Error code parameter not valid.
CPF3C19 E	Error occurred with receiver variable specified.
CPF3C20 E	Error found by program &1.
CPF3C21 E	Format name &1 is not valid.
CPF3C24 E	Length of the receiver variable is not valid.
CPF3C90 E	Literal value cannot be changed.
CPF35BE E	Product &1 &3 not supported or installed.
CPF358A E	Release not valid.
CPF3598 E	PTF function already in progress.
CPF3902 E	PTF &1 located more than once.
CPF6602 E	PTF &1-&2 &3 not found.
CPF9872 E	Program or service program &1 in library &2 ended. Reason code &3.

API introduced: V2R3

Top | Software Product APIs | APIs by category

Select Product (QSZSLTPR) API

Required Parameter Group:

1	Output List	Output	Array of Char(*)
2	Input information	Input	Char(40)
3	Format name	Input	Char(8)
4	Input list	Input	Array of Char(18)
5	Output information	Output	Char(12)
6	Error code	I/O	Char(*)

Default Public Authority: *USE

Threadsafe: No

The Select Product (QSZSLTPR) API displays a list of products. One or more products can then be selected from the prompt screen. The list of selected products is then returned to the caller of the API. The list of products displayed can be:

- All installed products.
- All supported products.
- All defined products.
- A user-specified subset of all defined products.
- All products that are supported, installed, or both installed and supported.

Alternatively, the QSZSLTPR API can return a list of products without displaying a list of products. The result is as if the list of products had been displayed and all products in the list had been selected.

If the job is in batch mode, the number of products to select field must be *ALL.

Note: A product can be supported and unsupported by using the Work with Supported Products (WRKSPTPRD) command. This command is part of the System Manager for iSeries licensed program.

A defined product is one which is known to the system. This includes all installed products, but also includes products which are known to the system without the products being installed. For example, V3R1M0 of the System Manager for iSeries licensed program (5763SM1) is known to the system once V3R1M0 of the operating system is installed. Therefore V3R1M0 of 5763SM1 is a defined product once V3R1M0 of the operating system is installed.

A product is also a defined product when a product definition (*PRDDFN) object exists for that product on the system.

Authorities and Locks

**PRDAVL Lock*

*SHRRD

Required Parameter Group

Output list

OUTPUT; Array of CHAR(*)

An array where you will receive one or more product records.

All the records of this array will have the same size. The size is returned in the record size field of the output information parameter.

When the number of products to select field of the input information parameter is:

**ONE* The output array must be able to contain at least one record or the results are unpredictable.

**ALL* The array contains a complete list of product ID records as if all product IDs were selected. No display is shown. If you do not provide enough space for all the records, as many records as can be returned in the space provided will be returned. The Records available field of the Output information parameter will indicate the number of records that were available to be returned.

**ANY* The array contains a list of product ID records selected from the display. It is assumed that you can select all of the product IDs shown on the display. Therefore, the output array has to be able to hold all the records shown on the display. This is checked before the display is shown. If the output array cannot hold all of the available records, an error message is returned and no display is shown.

For information about the layout of this parameter, see PRDS0100 Format and PRDS0200 Format.

Input information

INPUT; CHAR(40)

Specifies:

- The number of records to be returned.
- Whether a list of products is to be displayed for selection or whether no list is to be displayed.
- Several attributes of the display if the list is to be displayed for selection.

See Input Information Format.

Format name

INPUT; CHAR(8)

The content and format of the information returned.

The possible format names are:

PRDS0100 Basic information is returned. See PRDS0100 Format.

PRDS0200 This format includes more information than format PRDS0100. See PRDS0200 Format.

Input list

INPUT; Array of CHAR(18)

An array containing the records used to determine which products that are defined to the system are displayed or returned. The Input list is ignored unless the Product field is equal to *LIST. The format of the data for the input list is described in Input List Format.

Output information

OUTPUT; CHAR(12)

The following information is returned:

- The size of an output record.
- The number of records available.
- An indicator of what action was performed on the display.

See Output information Format.

Error code

I/O; CHAR(*)

The structure in which to return error information. For the format of the structure, see Error Code Parameter.

Input Information Format

Offset		Type	Field
Dec	Hex		
0	0	BINARY(4)	Number of records to return
4	4	CHAR(10)	Number of products to select
14	E	CHAR(1)	Initial panel view
15	F	CHAR(1)	Allow exit
16	10	CHAR(10)	Product options to display
26	1A	CHAR(10)	Product
36	24	BINARY(4)	Records in list

Field Descriptions for Input Information

When *LIST is specified for Product, then a list of product records must be specified for the Input List parameter.

Allow exit. Whether the exit action is allowed from the display.

The valid values are:

1 Exit and cancel are both allowed.

0 Only cancel is allowed.

Initial panel view. Whether the first view of the display shows the release level.

If the release level is shown on the first view, the description is not shown until the view is changed by pressing the appropriate key. If the first view does not show the release level, the appropriate key can be pressed to change the view and show the release level.

The valid values are:

1 The initial view of the display shows the release level.

2 The initial view of the display shows the description text.

Number of products to select. The number of products that are allowed to be selected on the display. This field controls whether a panel is displayed or not.

The valid values are:

**ONE* One product is allowed to be selected. If the job is in batch mode, CPF0C1A will be returned.

**ANY* Any number of products can be selected. If the job is in batch mode, CPF0C1A will be returned.

**ALL* No list is displayed; all the products will be selected.

Number of records to return. The number of records that can be put in the output list array.

Product. If the Number of products to select parameter is *ALL, this field specifies which products to return. Otherwise, this field specifies which products to put in the list of products to display.

The valid values are:

**INSTLD* Installed products. Installed means that a product definition (*PRDDFN) object and a product load (*PRDLOD) object for the code load were loaded onto the system by the Restore Licensed Program (RSTLICPGM) command.

**SUPPTD* Supported products. Products supported by using the Work with Supported Products (WRKSPTPRD) command which is part of the System Manager for iSeries licensed program.

**INSSPT* Installed or supported products. All products that are installed, supported, or both installed and supported.

**ALL* All known and defined products. This includes installed and supported and others that are not installed or supported.

**LIST* The list of products shown on the display or returned is derived from the list that is passed in the Input List array. The API builds a list of product ID, product option, release level combinations from the Input List parameter. See Field Descriptions for Input List for information.

Product options to display. Whether only the base option of each product is to be displayed or whether all options are to be displayed.

The valid values are:

**ALL* Show all options of a product

**BASE* Show only the base option of a product

Records in list. The number of records in the Input List array. This is used only when the Product field is equal to *LIST.

Output information Format

Offset		Type	Field
Dec	Hex		
0	0	BINARY(4)	Record size
4	4	BINARY(4)	Records available
8	8	BINARY(4)	Action

Field Descriptions for Output Information

Action. What action was performed on the display. If no display was shown, 0 is returned.

The possible values are:

-4 Exit

-8 Cancel

0 Enter

21 Select all

Records available. The number of records that are available to be returned.

If a panel is displayed and either the exit or cancel action is performed, (so no records were selected) this number is 0.

If no records are shown on the display, or no records are returned in the output array, this number will be equal to minus 1.

*ANY may be specified for the Number of products to select parameter. If so, in case of an error, this number indicates the necessary size of the output array. More storage is required for the API and no list is returned.

*ALL may be specified for the Number of products to select parameter. If so, this number indicates whether there were more records to return in the output array; a partial list may have been returned.

Record size. The length of the records of the output array.

- When format PRDS0100 is requested, this value is 83.
- When format PRDS0200 is requested, this value is 197.

Input list and Output list Formats

The output list parameter can be in one of two formats. See PRDS0100 Format, and PRDS0200 Format for information.

Input List Format

Offset		Type	Field
Dec	Hex		
0	0	CHAR(7)	Product ID
7	7	CHAR(5)	Product option
12	C	CHAR(6)	Release level

Field Descriptions for Input List

When *LIST is specified for the Product parameter, then a list of product records must be specified for the Input List parameter. The API builds a list of product ID, product option, release level combinations from the Input List parameter. All product ID, product option, release level combinations defined to the system, which match a record in the Input List, are put into a list. The list is displayed or returned.

Any combination of the product ID, product option, and release level fields can be left blank. A blank field matches all values for that field. For example, a product ID and product option are specified. But if the release level field is left blank, all available releases of the specified product option are displayed or returned.

If the Product ID, Product option, and Release level are all blank, all available products at all release levels are displayed or returned.

Product ID. The product IDs to display or return. The product ID must be a valid product ID for a defined product or blank. If the API does not find a valid product ID from the defined products, an error is returned.

Product option. The product option to display or return. The product option must be blank or a valid product option must be specified. Valid values are *BASE (00000), 00001, 00002, 00003, and so on. Leading zeros are required.

Release level. The release level of the product. The release level must be either blank or in the format VxRxMy. Valid values for x are 0 through 9. Valid values for y are 0 through 9 and A through Z.

PRDS0100 Format

Offset		Type	Field
Dec	Hex		
0	0	CHAR(7)	Product ID
7	7	CHAR(5)	Product option
12	C	CHAR(6)	Release level

18	12	CHAR(2)	Reserved
20	14	CHAR(7)	Description text message ID
27	1B	CHAR(10)	Description text object name
37	25	CHAR(46)	Description text

PRDS0200 Format

Offset		Type	Field
Dec	**Hex**		
0	0	CHAR(7)	Product ID
7	7	CHAR(5)	Product option
12	C	CHAR(6)	Release level
18	12	CHAR(2)	Reserved
20	14	CHAR(7)	Description text message ID
27	1B	CHAR(10)	Description text object name
37	25	CHAR(10)	Description text library name
47	2F	CHAR(1)	Installed flag
48	30	CHAR(1)	Supported flag
49	31	CHAR(2)	Registration type
51	33	CHAR(14)	Registration value
65	41	CHAR(132)	Description text

Field Descriptions for output list

Description text. Text in the message that describes the product option selected. To retrieve the message text, the library list is searched for the specified message file. If the message file is not found, then the description text library name is used to try to retrieve the message text. If the message file is not found, blanks are returned.

Note: This field is only 46 characters long for format PRDS0100 because product descriptions typically should not be longer than 46 characters. Product descriptions should not be longer than 46 characters because some system commands display a maximum of 46 characters for each product description.

Description text library name. The name of the library for the message file that contains the messages which describe the product and its options.

Description text message ID. A seven character alphanumeric identifier assigned to the message that describes the product option selected. The message ID for the base option is the message ID for the product.

Description text object name. The name of the message file that contains the messages which describe the product and its options.

Installed flag. Whether the code load for this product option is installed or not. A load is installed if a product load (*PRDLOD) object is loaded on the system by the Restore Licensed Program (RSTLICPGM) command.

The possible values are:

1 The code load is installed.

0 The code load is not installed.

Product ID. The product ID selected.

Product option. The product option of the product selected.

Registration type. The registration type associated with the product. The registration type and registration value together make up the registration ID for the product.

Registration value. The registration value associated with the product. The registration type and registration value together make up the registration ID for the product.

Release level. The release level of the product selected.

Reserved. This field will contain blanks.

Supported flag. Whether this load is currently supported. A load can be supported by using the Work with Supported Products (WRKSPTPRD) command in the System Manager licensed program.

The possible values are:

1 The load is supported.

0 The load is not supported.

Error Messages

Message ID	Error Message Text
CPF0C10 E	Input information parameter at offset &3 is not valid.
CPF0C11 E	Products listed in Input List array not found.
CPF0C12 E	Output List array too small.
CPF0C15 E	Error occurred while processing QSZSLTPR API.
CPF0C1A E	Panel could not be displayed.
CPF24B4 E	Severe error while addressing parameter list.
CPF3CF1 E	Error code parameter not valid.
CPF3C21 E	Format name &1 is not valid.
CPF3C90 E	Literal value cannot be changed.
CPF6A00 E	All CPF6Axx messages could be returned. xx is from 01 to FF.
CPF9872 E	Program or service program &1 in library &2 ended. Reason code &3.

API Introduced: V3R1

Update OS/400 Registered Application Information Repository (QSZUPDRA, QszUpdRegAppInfoRepository) API

Required Parameter Group:

1	Application information path name	Input	Char(*)
2	Function	Input	Binary(4)
3	Error code	I/O	Char(*)

Service Program Name: QSZRAIRA

Default Public Authority: *EXCLUDE

Threadsafe: No

The Update OS/400 Registered Application Information Repository (QSZUPDRA, QszUpdRegAppInfoRepository) API updates information about one or many separately installable pieces of an application called components. Each piece of information about each component is stored as a tag and a value. This API can be used to:

- Define a new component.
- Update a tag defined for a component.
- Add a new tag for the component.
- Remove a tag from the component.
- Remove a component and all its associated information (tags).

The flexibility of this API allows it to define all information for a component or a set of components in a single call since any number of components and tags may be specified in the same call.

It also can be used to remove a set of tags for a given component or it can even remove a set of components.

The user must be aware that this API should not be used to update information that belongs to OS/400 packaged products including the OS/400 Operating System, that information in the OS/400 Registered Application Information Repository is automatically updated by the system when a user installs an OS/400 packaged product using the Restore License Program (RSTLICPGM) command or when the user deletes the product using the Delete License Program (DLTLICPGM) command. If the user tries to updated information for an OS/400 packaged product, an error message will be signalled.

Authorities and Locks

Library Authority

*EXECUTE

Authority for user space containing XML document

*USE

User Space Lock

*EXCLRD

Stream File Directory Authority

*RX

Authority for Stream File containing XML document

*R

Required Parameter Group

Application information path name

INPUT; CHAR(*)

The path name of the object which contains the XML document with the information of the components to be updated by the API, this may be a path to a user space (*USRSPC) or a path to an stream file(*STMF). The path name should be specified in the Qlg_Path_Name_T format. If a pointer is specified in the path name format, it must be 16-byte aligned. If not, unpredictable results may occur. For more information on this structure, see Path name format. The information contained in this object should be given to the API in XML format according to the data type definition (DTD), for a detailed description of the DTD, see for a detailed description of the information contained in this object, see "XML Document when updating".

Depending on the function requested, some of the tags may be optional. For example, to remove all information related to a specific component, you have to specify the remove function and only the component information; that is, product name, version, component name, instance, feature and vendor. The API will remove the component from the repository along with its associated information.

Function

INPUT; BINARY(4)

The function to perform. The possible function values are:

1. The Add or Update function. It adds or updates information in the OS/400 Registered Application Information Repository. The following actions may occur in the same call:
   - If the component specified does not exist, it adds the component and all the tag(s) associated to it.
   - If the component already exists in the repository but its associated tags do not exist, the tags will be added.
   - If the component exists and its tags already exist, it updates the tags with their new values.
2. The remove function. It removes the tag or tags specified corresponding to the component. If no tags are specified for the component, the component and all its associated information is removed.

Error code

I/O; CHAR(*)

The structure in which to return error information. For the format of the structure, see Error Code

Parameter.

XML Document when updating

This object may be a stream file (*STMF) or a user space (*USRSPC) object. In either case, the contents of the object must be an XML document that conforms to the specified rules.

All elements and attributes defined in the data type definition(DTD) are allowed when using this API. See Software Components DTD for a detailed description of each element. The examples below illustrate the way to define an XML document to call this API.

Register only the component with no additional data.

When a component is being registered, use function add/update (1).

```
<?xml ComponentVersion="1.0" encoding='ucs-2'?>
<!DOCTYPE RegAppInfoRepository SYSTEM
"/IBM/XML/DTD/RegAppInfoRepository.dtd">
<RegAppInfoRepository DTDVersion="1.0">
  <Component ProductName="My compiler" ComponentVersion="v1.1.0"
ComponentName="Tools"
             ComponentVendor="Juan Perez" FeatureName="Juan Perez">
  </Component>
</RegAppInfoRepository>
```

Use one call to register and store all information related to a component.

```
<?xml ComponentVersion="1.0" encoding='ucs-2'?>
<!DOCTYPE RegAppInfoRepository SYSTEM
"/IBM/XML/DTD/RegAppInfoRepository.dtd">
<RegAppInfoRepository DTDVersion="1.0">
  <Component ProductName="My tools" ComponentVersion="v1.1.0"
ComponentName="Edit"
             Instance="/ProgramFiles/MyTools" ComponentVendor="Juan Perez">
    <ExtendedData UninstallInfo="java /ProgramFiles/MyTools/MyUninstaller"
       LastFixPackApplied="2.0" InstallerType="ISJE" Supported="0"
       Installed="1" CCSID="00037">
       <Shared>
          <SharingComponent ProductName="My compiler"
ComponentVersion="v1.1.0"
                            ComponentName="EditTools" ComponentVendor="Pedro
Fernandez"/>
       </Shared>
       <Files>
          <Directory
DirectoryName="/ProgramFiles/MyTools/MyUninstaller/bin">
             <FileName>edit1.exe</FileName>
             <FileName>edit2.exe</FileName>
             <FileName>edit3.exe</FileName>
```

```
            <FileName>edit4.exe</FileName>
            <FileName>edit5.exe</FileName>
          </Directory>
        </Files>
        <AdditionalValue ValueName="InstallType" Value="Custom"/>
        <AdditionalValue ValueName="InstallDate" Value="05/10/1999"/>
    </ExtendedData>
  </Component>
</RegAppInfoRepository>
```

Error Messages

Message ID	Error Message Text
CPF24B4 E	Severe error addressing parameter list.
CPF3CF1 E	Error code parameter not valid.
CPF3CF2 E	Error(s) occurred during running of &1 API.
CPF9872 E	Program or service program &1 in library &2 ended.
CPF3C1E E	Required parameter &1 omitted.
CPF0CC1 E	Error initializing the XML parser.
CPF0CC2 E	Error found on XML input document.
CPF0CC7 E	Requested function not successful.
CPF0CC6 E	Not all components were successfully updated.
CPF0CC4 E	Value &1 for function parameter not valid.
CPF0CD2 E	Application information path name not valid.

API introduced: V5R1

Program Temporary Fix Exit Program

Required Parameter Group:

1	Product ID	Input	Char(7)
2	PTF ID	Input	Char(7)
3	Product release level	Input	Char(6)
4	Product option ID	Input	Char(4)
5	Product load ID	Input	Char(4)
6	PTF library	Input	Char(10)
7	User data	Input	Char(50)
8	Current PTF status	Input	Char(1)
9	PTF operation	Input	Char(1)

When a program temporary fix (PTF) is created, one or more PTF exit programs may be specified. The PTF exit programs are called when a PTF is temporarily applied, permanently applied, temporarily removed, or permanently removed. Exit programs are called at the end of the apply or remove processing. The pre-exit programs are called both before and after the apply or remove processing. The run option field of the exit programs parameter determines when the exit program is called. (Refer to Exit Programs Format.) Exit programs eliminate the need for you to manually carry out special instructions to install the PTF.

Shipping the same exit program in two PTFs for the same product causes one PTF to supersede the other. Avoid this by including the PTF exit program when the product is initially packaged. If a PTF exit program already exists in the product, then specify *OBJLIST for the value of the exit program type field (see Exit Programs Format). Another way to avoid unwanted superseding PTFs is to ship the exit program with a different temporary object name in each PTF. When the exit program is being shipped with the PTF, specify the value *PTF for the exit program type field (see Exit Program Format).

The interface between the Apply PTF (APYPTF) command or Remove PTF (RMVPTF) command and the exit program follows. Your exit program must have the following input parameters.

Product ID

INPUT; CHAR(7)

The software product that the PTF affects.

PTF ID

INPUT; CHAR(7)

The identification number of the PTF.

Product release level

INPUT; CHAR(6)

The release of the software product that the PTF is for in the format VxRyMz. Valid values for x and y are 0 through 9, and valid values for z are 0 through 9 and A through Z.

Product option ID

INPUT; CHAR(4)

The option of the software product that the PTF is for.

Product load ID

INPUT; CHAR(4)

The load of the software product that the PTF is for.

PTF library

INPUT; CHAR(10)

The name of the library in which the PTF objects were placed.

User data

INPUT; CHAR(50)

Any user data specified by the PTF originator to be passed to the exit program.

Current PTF status

INPUT; CHAR(1)

The current status of the PTF.

0 Loaded but not applied

1 Applied temporarily

PTF operation

INPUT; CHAR(1)

The change being made to the status of the PTF.

0 Remove temporarily

1 Apply temporarily

2 Apply permanently

3 Remove permanently

4 Pre-remove temporarily

5 Pre-apply temporarily

6 Pre-apply permanently

7 Pre-remove permanently

Error Messages

Message ID	Error Message Text
CPF3638 E	Exit program failed, but made no permanent changes.
CPF3639 E	Exit program failed and permanent changes have been made.

If one of these messages is sent, the apply PTF process fails and no following PTFs will be applied or removed. This means that if you are installing a cumulative PTF package when this failure occurs, the cumulative package will not be applied.

If an exit program sends escape message CPF3638, any other exit programs already called for that PTF are called again. This allows changes to be backed out. During the back-out operation, if all the exit programs called by the PTF run successfully, the PTF status does not change from what it was before the PTF operation was attempted. For example, the PTF was not applied and the PTF exit program failed while the PTF was being temporarily applied. The status of the PTF would then remain not applied. During the blackout, the current PTF status is set to the original status of the PTF. In the example above, the exit programs are called with a current PTF status of 0 (loaded but not applied) and a PTF operation of either:

- 0 (remove temporarily)
- 4 (pre-remove temporarily)

This message should only be issued if any changes made by the exit program were backed out.

When message CPF3639 is signaled, the PTF is marked damaged. This indicates that the PTF has been partially applied. Any objects contained in the PTF have already been replaced. Some of the exit programs may have completed successfully.

If an exit program signals any other escape messages, unpredictable results can be expected. Exit programs must monitor for all exceptions. Other error messages should be left in the job log for problem determination or should be sent as informational or diagnostic messages. Message CPF3569 can be used to indicate that a special handling program failed.

You can also send a completion message (CPC1214) if the exit program runs successfully.

For more information, see Creating a Program Temporary Fix exit program in the API examples.

Exit program introduced: V2R2

Top | Software Product APIs | APIs by category

QLPUSER Exit Program

Required Parameter Group:			
1	Tape device name	Input	Char(10)
Threadsafe: No			

The QLPUSER exit program is called during the automatic installation process. It can be used to restore products that are not listed on the Licensed Program menu. More information about the QLPUSER program is available in the Distributed Data Management book.

Required Parameter

Tape device name

INPUT; CHAR(10)

The tape device name that identifies the tape device to be used.

QLPUSER Exit Program Example

The following example is a control language (CL) program that:

- Sends an instruction to the operator at the target site system.
- Restores a library containing an application.
- Copies the command to start the application into the QGPL library.

Note: The &DEVICE parameter in the example is the name of your tape device.

```
PGM PARM(&DEVICE)
DCL VAR(&DEVICE) TYPE(*CHAR) LEN(10)
SNDUSRMSG MSG('Load the tape containing Application 1, then press Enter.')
RSTLIB SAVLIB(APP1) DEV(&DEVICE)
CRTDUPOBJ OBJ(STRAPP1) FROMLIB(APP1) OBJTYPE(*CMD) TOLIB(QGPL)
ENDPGM
```

Exit program introduced: V2R3

Software Product Functions Exit Program

Required Parameter Group:

1	Function being performed	Input	Char(10)
2	Before or after indicator	Input	Char(10)
3	Language ID	Input	Char(4)
4	Library that contains exit programs	Input	Char(10)
5	Library where product currently exists	Input	Char(10)
6	Library specified by creator of product	Input	Char(10)
7	Library where product will exist	Input	Char(10)
8	Root folder that product currently uses	Input	Char(12)
9	Root folder specified by creator	Input	Char(12)
10	Root folder that product will use	Input	Char(12)
11	Release level of product	Input	Char(6)
12	Release level of product being restored	Input	Char(6)
13	Array of missing objects and their symbolic object types	Input	Array of Char(20)
14	Number of missing objects	Input	Binary(4)
15	Array of missing folders	Input	Array of Char(63)
16	Number of missing folders	Input	Binary(4)
17	Array of offsets to missing directory information	Input	Array of Binary(4)
18	Number of missing directories	Input	Binary(4)
19	Number of product directories	Input	Binary(4)
20	Array of offsets to directory information where product currently exists	Input	Array(300) of Binary(4)
21	Array of offsets to directory information specified by creator of product	Input	Array(300) of Binary(4)
22	Array of offsets to directory information where product will exist	Input	Array(300) of Binary(4)

This exit program is specified when you create products that are restored, saved, deleted, and checked with the following commands:

- Restore Licensed Program (RSTLICPGM)
- Delete Licensed Program (DLTLICPGM)
- Save Licensed Program (SAVLICPGM)
- Check Product Option (CHKPRDOPT)

More information about creating your own product and using exit programs is available in the System Manager Use book.

Required Parameter Group

Exit programs must be written to accept the following parameters:

Function being performed

INPUT; CHAR(10)

The function being performed when this exit program is called. The values are:

**RSTCODE (restore code)*	Restore program objects when the RSTLICPGM command runs.
**SAVCODE (save code)*	Save program objects when the SAVLICPGM command runs.
**DLTCODE (delete code)*	Delete program objects when the DLTLICPGM command runs.
**CHKCODE (check code)*	Check program objects when the CHKPRDOPT command runs.
**RSTLNG (restore language)*	Restore language objects when the RSTLICPGM command runs.
**SAVLNG (save language)*	Save language objects when the SAVLICPGM command runs.
**DLTLNG (delete language)*	Delete language objects when the DLTLICPGM command runs.
**CHKLNG (check language)*	Check language objects when the CHKPRDOPT command runs.

Before or after indicator

INPUT; CHAR(10)

Whether the exit program is being called before or after the actual operation (restore process, save process, and so on). The values are:

**BEFORE*	The program is called before the actual operation.
**AFTER*	The program is called after the actual operation.

Language ID

INPUT; CHAR(4)

The 4-digit feature code identifier. For the *RSTCODE, *SAVCODE, *DLTCODE, and *CHKCODE values of the function being performed parameter, this value is blank because it only applies to the language functions.

Library that contains exit programs

INPUT; CHAR(10)

The library where exit programs for the product are located while the current function is being performed. You cannot assume this value will be either the library where the product currently exists or will exist.

Library where product currently exists

INPUT; CHAR(10)

If the product exists on the system when the exit program is called, this is the library that contains the product. Otherwise, this value is blank.

Library specified by creator of product

INPUT; CHAR(10)

The library that was specified on the Create Product Load (CRTPRDLOD) command or the Create Product Load (QSZCRTPL) API when you created the product.

Library where product will exist

INPUT; CHAR(10)

The library of the product after the product is restored. This value only applies when the function being performed is *RSTCODE and *RSTLNG. For other functions, this value is blank.

Root folder that product currently uses

INPUT; CHAR(12)

The root folder specified if the product is installed and has folders. It is only specified when calling the exit program for the principal library. Otherwise, this value is blank.

Root folder specified by creator

INPUT; CHAR(12)

The root folder that was specified on the CRTPRDLOD command or the QSZCRTPL API when the product was created. This only applies if the product has folders and is only specified when calling the exit program for the principal library.

Root folder that product will use

INPUT; CHAR(12)

The root folder specified only if the product is being restored and has folders. It is only specified when calling the exit program for the principal library. Otherwise, this value is blank.

Release level of product

INPUT; CHAR(6)

The version, release, and modification level of the product being restored, saved, deleted, or checked in the form VxRxMx.

Release level of product being restored

INPUT; CHAR(6)

The version, release, and modification level of the product being restored in the form VxRxMx. It is only specified if the product is being restored; otherwise, the value is blank.

Array of missing objects and their symbolic object types

INPUT; ARRAY of CHAR(20)

The list passed when the function being performed is *SAVCODE, *SAVLNG, *CHKCODE, or *CHKLNG. For *SAVCODE and *SAVLNG, the list will only be passed before the actual save operation. For *CHKCODE and *CHKLNG, the list will be passed after the check operation. The first 10 characters contain the object name, and the second 10 characters contain the object type.

Number of missing objects

INPUT; BINARY(4)

The number of missing objects in the previous array. This number is 0 when there are no objects in the list or when the function being performed is not *SAVCODE, *SAVLNG, *CHKCODE, or *CHKLNG. The maximum number is 499.

Array of missing folders

INPUT; ARRAY of CHAR(63) for each folder name

The list passed when the function being performed is *SAVCODE, *SAVLNG, *CHKCODE, or *CHKLNG. For *SAVCODE and *SAVLNG, the list will only be passed before the actual save operation. For *CHKCODE and *CHKLNG, the list will be passed after the check operation. This only applies if the product has folders and is only specified when calling the exit program for the principal library.

Number of missing folders

INPUT; BINARY(4)

The number of missing folders in the previous array. This number is 0 when there are no folders in the list or when the function being performed is not *SAVCODE, *SAVLNG, *CHKCODE, or *CHKLNG.

Array of offsets to missing directory information

INPUT; ARRAY of BINARY(4)

The offsets from the beginning of this array that point to the missing directory information. The offsets and missing directory information structures are passed when the function being performed is *SAVCODE, *SAVLNG, *CHKCODE, or *CHKLNG. For *SAVCODE and *SAVLNG, the list will only be passed before the actual save operation. For *CHKCODE and *CHKLNG, the list will be passed after the check operation. This only applies if the product has directories, and is only specified when calling the exit program for the principal library. For a description of the format, see Format of Missing Directory Information.

Number of missing directories

INPUT; BINARY(4)

The number of missing directories in the previous array. The maximum number is 300. This number is 0 when there are no missing directories in the list or when the function being performed is not *SAVCODE, *SAVLNG, *CHKCODE, or *CHKLNG.

Number of product directories

INPUT; BINARY(4)

The number of product directories in the next three arrays. This number is 0 when there are no product directories in any of the following lists. The maximum number is 300.

Array of offsets to directory information where product currently exists

INPUT; ARRAY(300) of BINARY(4)

The offsets from the beginning of this array that point to the current product directory information. When the function being performed is *RSTCODE, *RSTLNG, *SAVCODE, or *SAVLNG, this information is passed both before and after the actual restore or save operation. When the function being performed is *DLTCODE or *DLTLNG, this information is passed before the delete operation. When the function being performed is *CHKCODE or *CHKLNG, this information is passed after the check operation. This array will contain zeros in the following circumstances:

- There are no product directories.
- The function being performed is *RSTCODE or *RSTLNG and the product is not currently installed.

For a description of the structure that contains the product directory information, see Format of Product Directory Information.

Array of offsets to directory information specified by creator of product

INPUT; ARRAY(300) of BINARY(4)

The offsets from the beginning of this array that point to the directory information specified when the product was created. When the function being performed is *RSTCODE, *RSTLNG, *SAVCODE, or *SAVLNG, this information is passed both before and after the actual restore or save operation. When the function being performed is *DLTCODE or *DLTLNG, this information is passed before the delete operation. When the function being performed is *CHKCODE or *CHKLNG, this information is passed after the check operation. This array will contain zeros if there are no product directories. For a description of the structure that contains the product directory information, see Format of Product Directory Information.

Array of offsets to directory information where product will exist

INPUT; ARRAY(300) of BINARY(4)

The offsets from the beginning of this array that point to the directory information where the product will exist after the restore operation. When the function being performed is *RSTCODE or *RSTLNG, this information is passed both before and after the actual restore operation. The information is not passed for other functions. This array will contain zeros if there are no product directories. For a description of the structure that contains the product directory information, see Format of Product Directory Information.

Format of Missing Directory Information

The following table shows the format of the missing directory information.

Offset		Type	Field
Dec	Hex		
0	0	BINARY(4)	Full length of the missing directory
4	4	BINARY(4)	Home length of the missing directory
8	8	BINARY(4)	Missing directory's CCSID
12	C	BINARY(4)	Missing directory's index
16	10	CHAR(10)	Reserved space
26	1A	CHAR(480)	Missing directory name

Field Descriptions

Full length of the missing directory. The total length of the missing directory path name.

Home length of the missing directory. The root (changeable) part length of the missing directory path name.

Missing directory name. »The missing directory path name. The CCSID of this value is contained in the Missing Directory's CCSID field. The maximum missing directory name length is 240 characters (480 bytes). «

Missing directory's CCSID. The coded character set identifier of the missing directory.

Missing directory's index. A reference index into the array of offsets to directory information specified by the

creator of the product.

Reserved space. Internal reserved space.

Format of Product Directory Information

The following table shows the format for the product directory information structures.

Offset		Type	Field
Dec	Hex		
0	0	BINARY(4)	Full length of the product directory
4	4	BINARY(4)	Home length of the product directory
8	8	BINARY(4)	Product directory's CCSID
12	C	CHAR(14)	Reserved space
26	1A	CHAR(480)	Product directory name

Field Descriptions

Full length of the product directory. The total length of the product directory path name.

Home length of the product directory. The root (changeable) part length of the product directory path name.

Product directory name. »The product directory path name. The CCSID of this value is contained in the Product directory's CCSID field. The maximum product directory name length is 240 characters (480 bytes). «

Product directory's CCSID. The coded character set identifier of the product directory.

Reserved space. Internal reserved space.

Error Messages

Message ID	Error Message Text
CPD3DC8 E	&5 &6 in &7 not found for product &1 option &2 release &4.
CPD3DE7 E	Folder &6 not found for product &1 option &2 release &4.
CPF3D95 E	Exit program processing failed.
CPF3D98 E	Exit program processing found error in product.