IBM

eserver

iSeries

Problem Management APIs

Version 5 Release 3

eserver

iSeries

Problem Management APIs

Version 5 Release 3

Note

Before using this information and the product it supports, be sure to read the information in "Notices," on page 101.

Sixth Edition (August 2005)

This edition applies to version 5, release 3, modification 0 of Operating System/400 (product number 5722-SS1) and to all subsequent releases and modifications until otherwise indicated in new editions. This version does not run on all reduced instruction set computer (RISC) models nor does it run on CISC models.

Contents

Problem Management APIs

The problem management APIs offer you the ability to write problem management solutions, improve serviceability, and manage your own applications. Problem management APIs deal directly with how the iSeries(TM) server handles problems today. The problem log provides most of the operations necessary for problem management in a network environment.

For additional information on the problem management APIs, see:

- “Key Groups for Problem Log APIs” on page 57
- “Filtering” on page 2
- “Working with a Problem” on page 2

The problem management APIs are:

- “Add Problem Log Entry (QsxAddProblemLogEntry) API” on page 3 (QsxAddProblemLogEntry) adds additional or supporting data to a problem log entry.
- “Change Problem Log Entry (QsxChangeProblemLogEntry) API” on page 7 (QsxChangeProblemLogEntry) updates an existing problem entry by changing the information.
- “Create Problem Log Entry (QsxCreateProblemLogEntry) API” on page 14 (QsxCreateProblemLogEntry) creates a problem log entry with the information provided to the problem log entry.
- “Delete Problem Log Entry (QsxDeleteProblemLogEntry) API” on page 19 (QsxDeleteProblemLogEntry) deletes problem log entries or removes keys from a problem log entry.
- “End Problem Log Services (QsxEndProblemLogServices) API” on page 22 (QsxEndProblemLogServices) ends an instance of the problem log services identified by the handle returned when the services started.
- “Filter Problem (QSXFTRPB) API” on page 23 (QSXFTRPB) applies the currently active problem log filter to a problem log entry.
- “Log Software Error (QPDLOGER) API” on page 25 (QPDLOGER) logs a software problem and collects data needed for its resolution.
- “Report Software Error (QpdReportSoftwareError) API” on page 29 (QpdReportSoftwareError) is an ILE program that logs problems in the problem log and sends it to a service provider.
- » “Retrieve Policy Data (QPDETRTV) API” on page 39 (QPDETRTV) retrieves policy data«
- “Retrieve Problem Log Entry (QsxRetrieveProblemLogEntry) API” on page 41 (QsxRetrieveProblemLogEntry) extracts data from a specific problem log entry.
- “Retrieve Service Attributes (QESRSRVA) API” on page 44 (QESRSRVA) retrieves service information such as the service provider and whether automatic problem analysis should be performed.
- » “Set User Policy (QPDETPOL) API” on page 50 (QPDETPOL) allows the changing of user policies related to service.«
- “Start Problem Log Services (QsxStartProblemLogServices) API” on page 52 (QsxStartProblemLogServices) sets up an environment for adding, creating, changing, deleting, and retrieving problem log entries.
- “Work with Problem (QPDWRKPB) API” on page 54 (QPDWRKPB) analyzes and prepares a machine-detected hardware problem for reporting.

The problem log entry APIs are used to design problem log applications. Those APIs are:

- Add Problem Log Entry (QsxAddProblemLogEntry)
- Change Problem Log Entry (QsxChangeProblemLogEntry)
- Create Problem Log Entry (QsxCreateProblemLogEntry)

- Delete Problem Log Entry (QsxDeleteProblemLogEntry)
- End Problem Log Services (QsxEndProblemLogServices)
- Retrieve Problem Log Entry (QsxRetrieveProblemLogEntry)
- Start Problem Log Services (QsxStartProblemLogServices)

The error reporting APIs log software problems. Those APIs are:
- Log Software Error (QPDLOGER)
- Report Software Error (QpdReportSoftwareError)

Filtering

In the problem management APIs, a **filter** categorizes problem log entries into groups and performs operations on them accordingly.

The problem log applies the currently active filter to a problem log entry whenever a problem entry is created, changed, or deleted using system-provided interfaces.

The operations supported allow you to send application notification to a user data queue and assign the problem to a user. Your application can receive these notifications from the data queue using existing APIs.

See also Data Queue APIs.

Working with a Problem

Problem analysis is the process of finding the cause of a problem and identifying why the system is not working. Often, this process identifies equipment or data communications functions as the source of the problem. The "Work with Problem (QPDWRKPB) API" on page 54 (QPDWRKPB) API allows you to perform problem analysis on local machine-detected problems in the problem log. The Work with Problem (QPDWRKPB) API prepares the problem in the problem log for reporting; it does not report the problem automatically.

Top | APIs by category

APIs

These are the APIs for this category.

Add Problem Log Entry (QsxAddProblemLogEntry) API

Required Parameter Group:

1	Handle	
Input	Binary(4)	
2	Key structures	
Input	Array of Pointers	
3	Number of keys	
Input	Binary(4)	
4	Error code	
I/O	Char(*)	

Default Public Authority: *USE

Service Program: QSXSRUPL

Threadsafe: No

The Add Problem Log Entry (QsxAddProblemLogEntry) API adds information to an existing problem log entry.

The API supports the following data types:

- Keys 2001-2009 (field replaceable unit (FRU) entries) can be added to the problem log entry.
- Keys 4001 and 4002 (supporting data) entries can be added. Do not add duplicate information because checking is not performed.
- Key 6001 (history information) can be added.
- Key 7001 (PTF ID) can be added to a problem log entry. If the PTF entry already exists, an error is signalled.

Authorities and Locks

API Public Authority
*USE

Required Parameter Group

Handle
INPUT; BINARY(4)

An identifier that associates the problem log services started by the Start Problem Log Services API.

Key structures
INPUT; ARRAY of POINTERS

An array of pointers that has the address of each key that contains data to be written into the problem log. The number of pointers passed in the array must equal the value passed by the Number of keys parameter. Keys not supported for the Add Problem Log Entry API cause error messages to be sent to the caller.

Number of keys
INPUT; BINARY(4)

Number of keys passed to the API.

Error code
I/O; CHAR(*)

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

Rules for Key Usage

Key 1 (problem log ID) is required to identify the problem log entry to process.

Data can be added to an existing problem log entry with the Add Problem Log Entry API. The types of data that may be added with this API are:

- Keys 2001-2009 (FRU entries)
- Supporting data entries (keys 4001 and 4002 (supporting data)
- Key 6001 (history information)
- Key 7001 (PTF ID)

The remaining data contained in a problem can be altered using the Change API. More information on the above keys can be found in “Key Groups for Problem Log APIs” on page 57

Keys for Adding FRU Records

A FRU, field replaceable unit, entry defines an object that may have a specific machine-detected problem. FRUs have been broken into 9 types and represented by keys 2001 through 2009.

The types are:

2001	Device FRU type
2002	Code FRU type
2003	Media FRU type
2004	User FRU type
2005	FRU name
2006	Attached FRU
2007	Configuration FRU
2008	General FRU
2009	Channel attached FRU

In addition, a FRU or list of FRUs are associated with a problem based upon an analysis class. The analysis class implies the amount or type of analysis that has been done on the problem. FRUs are associated with a problem within the context of a class.

The classes of FRUs are:

1	Point of failure
2	Partial isolation
3	Isolation
4	Verification
5	Recovery
6	Answer

To add FRUs for a class of FRUs, the problem log entry must be identified, the class must be chosen, and the data must be added. These three actions need to be done for each FRU type. FRUs may be used in any combination, to add data about individual failing elements to a maximum of 21 FRUs per class.

This API adds FRU entries to the bottom of the list. If they need to be maintained in probability order, follow these steps:

- Retrieve the group using the Retrieve Problem Log Entry API.
- Modify the FRU records or append additional FRUs to the original list.
- Delete the existing FRU entries of that class using the Delete Problem Log Entry API.
- Add the new or updated FRU list using the Add Problem Log Entry API.

Keys for Adding Supporting Data

The addition of supporting data is not restricted. Any number of spooled or data base files can be associated with a problem log entry. Duplicate records are allowed. If you add a file twice, it is listed twice.

To add supporting data, define the type of record to be added using keys 4001 and 4002 (supporting data). They can be added in any combination.

Keys for Adding History Data

The addition of history data, or events, is restricted because specific events can occur only when the problem is in a specific status. Some history data types are applicable to specific problem log types. Any number of events can be associated with a problem log entry. Duplicate records are allowed since many events can be repeated. Events are added in the sequence that you supply them on the API call. The API makes no attempt to put them in order.

To add a history entry, use key 6001 (history information) to supply the needed data to reflect the action that was taken. If you are adding supporting data, you can add it in any combination. The time the event is added is entered by the API.

The history types are:

0	Problem entry closed. Only applicable when the problem has been closed. Once this status is set, the problem can only be retrieved.
1	Problem entry opened. Can only be used when the problem is initially opened.
2	Service request received. Only applicable when a problem is received from another system.
3	Opened by an alert. Only applicable when the problem is opened due to an alert.
4	Problem analyzed. Applicable each time a problem is analyzed.
5	Verification test ran. Applicable each time a problem is verified.
6	Recovery procedure ran. Applicable each time recovery is run.
7	Prepared to report. Applicable each time a problem is prepared to be sent to a service provider.
8	Service request sent. Applicable only when a problem is sent to another system. This implies that the service request was sent, but the service provider has no solution to the problem.
9	Problem answered. Applicable only when a problem is sent to another system. This implies that the service request was sent, and the service provider has a solution to the problem.
10	Response sent. Implies that a reply has been received from a service provider.
11	Reported by voice. Used when a problem is reported manually.
12	Fixes transmitted. Implies that fixes have been sent to a service requester.
13	A change request was submitted for this problem.
14	The change request submitted for this problem has ended.
15	Fix verified. Applicable each time a problem is verified.
16	Remote analysis. Only applicable when a problem has been analyzed by a remote service representative.
17	Remote verification ran. Only applicable when this system has been used to analyze a problem on another system.
18	Remote recovery ran. Only applicable when this system has been used to perform recovery on another system.
19	Alert created. Only applicable when the system created an alert for this problem.
20	APAR created. Only applicable when APAR data is created during analysis.
21	APAR data collected. Only applicable when APAR data is collected during analysis.

22	APAR data restored. Only applicable when APAR data is restored during analysis.
23	APAR data deleted. Only applicable when APAR data is deleted during analysis.
24	Changed by CHGPRB. Only applicable when the problem was changed by the CHGPRB command or the QsxChangeProblemLogEntry API.
25	Deleted by DLTPRB. Only applicable when the problem was changed by the DLTPRB command or QsxDeleteProblemLogEntry API.
26	This problem has occurred multiple times.
27	Status changed. Only applicable when querying the status of a problem that has been reported to a service provider.
28	Status query sent. Only applicable when querying the status of a problem that has been reported to a service provider.
29	Automatic problem analysis has completed successfully.
30	Auto-PAR is not complete; the SRC flag is off. Problem analysis did not occur because the SRC was turned off.
31	Auto-PAR not complete, submit job to QSYSWRK failed.
32	Auto-PAR failed. Problem analysis failed because an unknown problem occurred.
33	Auto-Notify complete. Problem was sent automatically.
34	Auto-Notify not complete, SRC flag is off. Problem was not sent automatically, the SRC was turned off.
35	An attempt to automatically send the problem failed.
36	Auto-Notify failed.
37	Problem analysis failed.

See Getting started with iSeries for more information about SRCs.

Keys for Adding PTF Entry

PTF entries can be added to the problem log at any time. Duplicate PTF records are not allowed and signal an exception condition. To ensure uniqueness, the PTF identifier and the product data are required.

To add a PTF record, use key 7001 (PTF ID) to add the data required. The PTF entry is added to the bottom of existing text.

To get the PTF records in a specific order, the records must be retrieved, sorted and then replaced after the existing PTF records are deleted.

PTF entries can be created using *ONLY*PRODUCT** as the constant for Product ID and *ONLY as the constant for version, release, and modification level.

Error Messages

Message ID	Error Message Text
CPF3C1E E	Required parameter &1 omitted.
CPF3C90 E	Literal value cannot be changed.
CPF3CF1 E	Error code parameter not valid.
CPF3CF2 E	Error(s) occurred during running of &1 API.
CPF7AAB E	Problem &1 not found.
CPF3C4D D	Length &1 for key &2 not valid.
CPF3C82 D	Key &1 not valid for API &2.
CPF3C86 D	Required key &1 not specified.
CPD7A82 D	Value not valid for key &1. (char string)
CPD7A83 D	Value not valid for key &1. (integer)
CPD7A88 D	Incorrect DBCS field format found.
CPD7A8A D	Key value &1 is not valid.
CPF7A89 E	Incorrect handle for this activation.
CPF7A8A E	Problem log services not started.

Message ID	Error Message Text
CPF7AA7 E	Problem &1 not found.
CPF9821 E	Not authorized to program &1 in library &2.
CPF9872 E	Program or service program &1 in library &2 ended. Reason code &3.
CPFA320 E	Pointer parameter is null.

API introduced: V3R1

Top | "Problem Management APIs," on page 1
APIs by category

Change Problem Log Entry (QsxChangeProblemLogEntry) API

Required Parameter Group:

1	Handle
Input	Binary(4)
2	Key structures
Input	Array of Pointers
3	Number of keys
Input	Binary(4)
4	Error code
I/O	Char(*)

Default Public Authority: *EXCLUDE

Service Program: QSXSRVPL

Threadsafe: No

The Change Problem Log Entry (QsxChangeProblemLogEntry) API updates an existing problem entry by changing the information. Key 1 (problem log ID) identifies the problem to be changed. Some data in the problem log entry can be changed on a field by field basis while other data can only be changed as a group and some data cannot be changed.

Authorities and Locks

API Public Authority
*USE

Required Parameter Group

Handle
INPUT; BINARY(4)

An identifier that associates the problem log services started with the QsxStartProblemLogServices API.

Key structures
INPUT; ARRAY of POINTERS

An array of pointers to the key structures being passed to the API.

Number of keys
INPUT; BINARY(4)

Number of keys passed to the API.

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 Keys

The number of keys used varies depending on the type of problem log entry being changed. You must select the keys applicable to the problem type with which you are working. If the keys provided to the API do not match the requirements for the problem log entry type you are changing, you are notified by the error handling procedures.

For details about the keys that can be used, see "Key Groups for Problem Log APIs" on page 57.

Rules for Key Usage

You can change the problem log data, the status, or both. The problems are categorized into the following types:

1	Machine-detected problem
2	User-perceived hardware or software problem
3	PTF orders
4	User-perceived remote problem
5	Application-detected problem
6	Client machine-detected problem
7	Client user-detected problem
8	User-created general problem

Changing General Data

General data is data that can be changed for any problem type without affecting the status of the problem. Data of this class are:

- Key 4 (user assigned). The validity of this data is not checked.
- Key 3001 (text entry) problem types 1, 2, and 4 can be changed.
- Key 6001 (history information). Use the Add Problem Log Entry (QsxAddProblemLogEntry) API.

Changing Problem Status

To change the problem status, specific data is required. The amount of data depends on the current or requested problem log status. Data that requires a status change cannot be added unless key 3 (problem status) is provided. An error is signaled if this occurs.

A problem can be changed to the following statuses:

- OPENED
  The beginning status of a problem.
- OPENED-PREPARED
  Problem is staged for transmission to a service provider.
- READY
  Problem entry has been analyzed and data is provided by keys that reflects the analysis results.
- READY-PREPARED
  Problem is staged for transmission to a service provider.

- SENT

  Problem has been sent to a service provider and a solution was not available.
- SENT-PREPARED

  Problem is staged for transmission to a service provider.
- ANSWERED

  Problem has been sent to a service provider and a solution was available.
- ANSWERED-PREPARED

  Problem is staged for transmission to a service provider.
- VERIFIED

  User has applied and tested the solution provided. The results of the testing are satisfactory. Once a problem is moved to VERIFIED status, it cannot be returned to OPENED or READY status.
- VERIFIED-PREPARED

  Problem is staged for transmission to a service provider.
- CLOSED

  Problem is resolved and there is no longer a need for the problem entry. Once this status is set, it cannot be returned to any other status. The problem can only be retrieved.

PREPARED, while displayed as a specific status, is actually an amplifier to the previous status of the problem: OPENED, READY, SENT, ANSWERED or VERIFIED.

The supplemental data needed to move a problem to PREPARED status are:
- Key 6 (operational data)
  - Prepared for system

    Required to define the system that this problem will be sent to.
- Key 1001 (problem severity)
  - Optional. If not provided, the API defaults it to None.
  - Prohibited for PTF orders (problem type 3).
- Key 1011 (PTF media selection)

  Optional. Defines the media on which a program fix can be delivered. If not provided, the contact data is used as a default. Typically this is the tape device type and model or a description of the media type on which PTFs are delivered if the distribution size exceeds a predefined transmit size limit.
- Key 5001 (contact information)

  Optional. Used to override local service contact information.

A problem can be set to PREPARED status by providing the required data keys (if not already provided) and a key 3 (problem status) code of 5.

Keys for Changing Problem Type 1 to Another Status

Problem type 1 (machine-detected problems) requires data from two additional key groups, 1000 and 2000.

Note: When changing status and FRU entries are required, use the Add Problem Log Entry API. To change status in general, you do not need the key group 2000 data.
- Data for OPENED status

  A problem in OPENED status, with the exception of general data that does not affect a status change, can only be changed to READY status. A problem may be changed from OPENED status to PREPARED status if the problem is to be sent to an iSeries server that has System Manager installed.
- Data for READY status

  A problem can be changed from OPENED to READY status by including the following data items:

  - Key 3 (problem status) indicates READY status.
  - Key 1004 (reporting device) is always required to identify the product with a maintenance contract, regardless of the problem.
  - Key 1006 (reporting code) is required for problems that were, on analysis, determined to be caused by software.
  - Key 1010 (symptom string).
  - Key 1007 (problem analysis data).
- Data for SENT and ANSWERED status at the service requester

  A problem can be changed to SENT or ANSWERED status by including the following data items:
  - Key 3 (problem status) indicates SENT or ANSWERED status.
  - Key 8 (answer codes).

    At the service requester the answer code assigned field should be updated with the answer received from a service provider that is not *IBMSRV.

    If *IBMSRV was the service provider, the Problem number field should be updated to reflect the problem management number that *IBMSRV has associated with the problem.
  - Key 7001 (PTF ID) is used to change the Sent and Status fields.

  Once the problem is in SENT or ANSWERED status, text can be added that defines the problem status. This is data that is added as a response to a query of the problem status or as an answer the service provider sends to the problem. This is done with key 3001 (text entry), type 3. Key 6001 (history information) is required to indicate the action has taken place.

  If the local system is acting as a service provider, the problem log entry for the service requester can be updated to reflect the following conditions:
  - A response was sent
  - PTFs were sent
  - An answer was added to the problem log

  These actions do not require a status change. Add a history event to define the action taken.
- Data for ANSWERED status at the service provider

  When a service provider answers a problem, the status is changed from READY status to ANSWERED status by including the following data items:
  - Key 3 (problem status) indicates ANSWERED status.
  - Key 6001 (history information) can add a number of events depending on the status change.
  - Key 7001 (PTF ID) is used to change the Sent and Status fields of the PTF entry.

  Once the problem is in ANSWERED status, text can be added that defines the problem status. This is data that is added as a response to a query of the problem status or as an answer the service provider wants to add to the problem. This is done with key 3001 (text entry), type 3. Key 6001 (history information) is required to indicate the action has taken place.

  When a response is sent to the service requester, key 6 (operational data) is used to indicate that a response was sent. The following data items are required:
  - Key 8 (answer codes) is updated to indicate the answer that was sent to the service requester.
  - Key 6001 (history information) can add a number of events depending on the status change.
  - Key 7001 (PTF ID) is used to change the Sent and Status fields.

  If PTFs were sent or an answer was added to the problem log, these actions do not require a status change. Add a history event to define the action taken.
- Data for VERIFIED status

  A problem can be changed to VERIFIED status from READY, SENT or ANSWERED status by including the following data items:
  - Key 3 (problem status) indicates VERIFIED status.
  - Key 1008 (fix verification status) where the Status field and PDP field must be provided.

  - Key 6001 (history information) can add a number of events depending on the status change. The Remote verification ran field is required based on the origin system location.
  - FRUs (key group 2000) are added for machine-detected problems and are added with the Add Problem Log Entry API.
- Data for recovery

  Recovery data can be added from OPENED or READY by including the following data items, and the status is not changed:
  - Key 3 (problem status) is not permitted. The status does not change as a result of running recovery procedures.
  - Key 1009 (fix recovery status) and problem determination procedures (PDP) fields must be provided.
  - Key 6001 (history information) can add a number of events depending on the status change. Remote recovery ran is required based on the origin system location.
- Data for CLOSED status

  A problem can be changed to CLOSED status from any other status by including the following data items:
  - Key 3 (problem status) indicates that CLOSED status is the only key allowed.
  - Key 6 (operational data) is updated by the API when the problem is closed.

Keys for Changing Problem Types 2, 4, 5, and 8

Problem types 2 (User-perceived), 4 (User-perceived remote), 5 (Application-detected), and 8 (User-created general) require the following data to achieve the following status:

- Data for OPENED status

  This does not apply because these problem types are created in READY status.
- Data for READY status

  This does not apply because these problems are created in READY status.
- Data for SENT and ANSWERED status at the service requester

  A problem can be changed to SENT or ANSWERED status by including the following data items:
  - Key 3 (problem status) indicates SENT or ANSWERED status.
  - Key 8 (answer codes).

    At the service requester, the Answer code assigned field should be updated with the answer received from a service provider that is not *IBMSRV.

    If *IBMSRV was the service provider, the problem management number (PMR) number field should be updated to reflect the problem management number that *IBMSRV has associated with the problem.
  - Key 6001 (history information) can add a number of events depending on the status change.
  - Key 7001 (PTF ID) is used to change the Sent and Status fields.

  Once the problem is in SENT or ANSWERED status, text can be added that defines the problem status. This is data that is added as a response to a query of the problem status or as an answer the service provider sends to the problem. This is done with key 3001 (text entry), type 3. Key 6001 (history information) is required to indicate the action has taken place.

  If the local system is acting as a service provider, the problem log entry for the service requester can be updated to reflect that a response was sent, PTFs were sent, or that an answer is added to the problem log. These actions do not require a status change. Add a history event to define the action taken.
- Data for ANSWERED status at the service provider

  When a service provider answers a problem, the status is changed from READY status to ANSWERED status by including the following data items:
  - Key 3 (problem status) indicates ANSWERED status.
  - Key 6001 (history information) can add a number of entries depending on the status change.
  - Key 7001 (PTF ID) is used to change the Sent and Status fields.

Once the problem is in ANSWERED status, text can be added that defines the problem status. This is data that is added as a response to a query of the problem status or as an answer the service provider wants to add to the problem. This is done with key 3001 (text entry), type 3. Key 6001 (history information) is required to indicate the action has taken place.

When a response is sent to the service requester, the problem log entry is updated to reflect that a response was sent. The following data items are required:

– Key 8 (answer codes) is updated to indicate the answer that was sent to the service requester.
– Key 6001 (history information) can add a number of events depending on the status change.
– Key 7001 (PTF ID) is used to change the Sent and Status fields.

If PTFs were sent or an answer was added to the problem log, these actions do not require a status change. Add a history event to define the action taken.

- Data for CLOSED status

  A problem can be changed to CLOSED status from any other status by including the following data items:

  – Key 3 (problem status) indicates that CLOSED status is the only key allowed.
  – Key 6 (operational data) is updated by the API when the problem is closed.

Keys for Changing Problem Type 3

Problem type 3 (PTF order) requires the following data to achieve the following status:

- Data for OPENED status

  This does not apply to PTF orders (problem type 3).

- Data for READY status

  This does not apply to PTF orders (problem type 3) because they are created in READY status.

- Data for SENT and ANSWERED status at the service requester

  A problem can be changed to SENT or ANSWERED status by including the following data items:

  – Key 3 (problem status) indicates SENT or ANSWERED status.
  – Key 8 (answer codes).

    At the service requester the Answer code assigned field should be updated with the answer received from a service provider that is not *IBMSRV.

    If *IBMSRV was the service provider, the Problem number field should be updated to reflect the problem management number that *IBMSRV has associated with the problem.

  – Key 6001 (history information) can add a number of events depending on the status change.
  – Key 7001 (PTF ID) is used to change the Sent and Status fields. This key is also used to update the product and VRM fields of the PTFs, especially if the default product, *ONLYPRD, and version, *ONLY, were used during the creation of the problem or if the SNDPTFORD command used the defaults.

  Once the problem is in SENT or ANSWERED status, text can be added that defines the problem status. This is data that is added as a response to a query of the problem status or as an answer the service provider sends to the problem. This is done with key 3001 (text entry), type 3. Key 6001 (history information) is required to indicate the action has taken place.

  If the local system is acting as a service provider, the problem log entry for the service requester can be updated to reflect that a response was sent, PTFs were sent, or that an answer is added to the problem log. These actions do not require a status change. Add a history event to define the action taken.

- Data for ANSWERED status at the service provider

  When a service provider answers a problem, the status is changed from READY status to ANSWERED status by including the following data items:

  – Key 3 (problem status) indicates ANSWERED status.
  – Key 6001 (history information) can add a number of events depending on the status change.
  – Key 7001 (PTF ID) is used to change the Sent and Status fields.

Once the problem is in ANSWERED status, text can be added that defines the problem status. This is data that is added as a response to a query of the problem status or as an answer the service provider wants to add to the problem. This is done with key 3001 (text entry), type 3. Key 6001 (history information) is required to indicate the action has taken place.

When a response is sent to the service requester, the problem log entry is updated to reflect that a response was sent. The following data items are required:

– Key 8 (answer codes) is updated to indicate the answer that was sent to the service requester.
– Key 6001 (history information) can add a number of events depending on the status change.
– Key 7001 (PTF ID) is used to change the Sent and Status fields.

If PTFs were sent or an answer was added to the problem log, these actions do not require a status change. Add a history event to define the action taken.

• Data for CLOSED status

A problem can be changed to CLOSED status from any other status by including the following data items:

– Key 3 (problem status) indicates that CLOSED status is the only key allowed.
– Key 6 (operational data) is updated by the API when the problem is closed.

Keys for Changing Problem Types 6 and 7

Problem type 6 (client machine-detected) and problem type 7 (client user-detected) require the Problem category field in key 1012 be set to 0 (Report) to move to PREPARED status.

Error Messages

Message ID	**Error Message Text**
CPF3C1E E	Required parameter &1 omitted.
CPF3C90 E	Literal value cannot be changed.
CPF3CF1 E	Error code parameter not valid.
CPF3CF2 E	Error(s) occurred during running of &1 API.
CPF7AAB E	Problem &1 not found.
CPF3C4D D	Length &1 for key &2 not valid.
CPF3C82 D	Key &1 not valid for API &2.
CPF3C86 D	Required key &1 not specified.
CPD7A82 D	Value not valid for key &1. (char string)
CPD7A83 D	Value not valid for key &1. (integer)
CPD7A87 D	Key &1 may be added only once.
CPD7A88 D	Incorrect DBCS field format found.
CPD7A8A D	Key value &1 is not valid.
CPD7A8B D	Length of data not valid.
CPF7A89 E	Incorrect handle for this activation.
CPF7A8A E	Problem log services not started.
CPF7AA7 E	Problem &1 not found.
CPF9821 E	Not authorized to program &1 in library &2.
CPF9872 E	Program or service program &1 in library &2 ended. Reason code &3.
CPFA320 E	Pointer parameter is null.

API introduced: V3R1

Top | “Problem Management APIs,” on page 1
APIs by category

Create Problem Log Entry (QsxCreateProblemLogEntry) API

Required Parameter Group:

1	Handle	
Input	Binary(4)	
2	Problem log ID	
Output	Char(40)	
3	Key structures	
Input	Array of Pointers	
4	Number of keys	
Input	Binary(4)	
5	Error Code	
I/O	Char(*)	

Service Program Name: QSXSRVPL

Default Public Authority: *USE

Threadsafe: No

The Create Problem Log Entry (QsxCreateProblemLogEntry) API creates a problem log entry and adds the information provided to the problem log files using keys. Key 1 (problem log ID) is returned to the user that is required for other API operations.

The API allows a problem to be created with a status of OPENED, READY, or PREPARED. The difference to the user is that the amount of data increases as the problem goes from OPENED to PREPARED.

The types of problems that may be created are:
- Machine-detected (problem type 1)
- User-perceived (problem type 2)
- PTF orders (problem type 3)
- User-perceived remote hardware or software problems (problem type 4)
- Application-detected (problem type 5)
- Client machine-detected (problem type 6)
- Client user-detected (problem type 7)
- User-created general (problem type 8)

The keys provided to create a problem are checked for validity and applicability to the problem log entry in two ways:
- Applicability of the type
- Applicability of the field

The fields are checked to verify that they are not null. Some keys allow the user to control them (key control). Keys without ″key control″ support require all fields to be filled. Fields not flagged are ignored.(The existence of the data is verified; NOT whether or not the data is valid. The problem log APIs do not check the validity of the data.) Operations that are unsupported or not valid, such as creating a problem in SENT status or not providing all dependent keys, result in a diagnostic message for each infraction found and an exception or an error notification defining it.

The key fields are checked before the problem log entry is created and an error is signalled if any required key fields are null.

If the maximum number of problem log entries has been reached for this particular date (99999), CPF392F E is signaled.

Authorities and Locks

API Public Authority
*USE

Required Parameter Group

Handle
INPUT; BINARY(4)

An identifier that associates the problem log services started by the Start Problem Log Services API.

Key 1 (problem log ID)
OUTPUT: CHAR(40)

This parameter contains the problem log identifier after the problem is created. If this parameter is omitted, no problem log ID is returned.

Key structures
INPUT; ARRAY of POINTERS

An array of pointers that has the address of each key that contains data to be written into the problem log. The number of pointers passed in the array must equal the value passed by parameter 4, Number of keys.

Number of keys
INPUT; BINARY(4)

Defines the number of keys that are being passed to the API.

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 Keys

See "Key Groups for Problem Log APIs" on page 57 for a description of the keys.

Rules for Key Usage

To create a problem log entry, specific data is required. The amount of data varies depending on the status of the problem log. This section defines the data required for each problem type for statuses OPENED and READY. Problems in OPENED or READY status may be amplified with the status of PREPARED. The supplemental data required for this is defined once and is applicable to either OPENED or READY.

General Keys For Problem Log Entry Data

The minimum key data for all problems types is:

- Key 1 (problem log ID)

This key must contain one of the following:

  - Problem log identifier if the problem was created on another system
  - The value ″*LOCAL″ to indicate that the problem is a local one.

- Key 2 (problem type)

  This defines the type of problem log entry being created.

- Key 3 (problem status)

  Defines the status to which the entry will be set. Three values are allowed:

  - OPENED. Create a problem in OPENED status.
  - READY. Create a problem in READY status.
  - PREPARED. Create a problem in READY or OPENED status and add supplemental data required for PREPARED status.

- Key 5 (problem origin system)
  - Required if key 1 (problem log ID) is not *LOCAL.
  - Prohibited if key 1 (problem log ID) is *LOCAL.

- Key 6 (operational data)
  - If key 1 (problem log ID) is *LOCAL, the Received from system field is prohibited. If it is not *LOCAL, the Received from system field is required.
  - If key 3 (problem status) indicates the problem is to be in PREPARED status the Prepared for system field is required otherwise it is prohibited.
  - The Date and time added field is automatically added by the API.
  - The Date and time closed field is automatically added when the problem is closed.

- Key 1002 (problem description message) or key 3001 (text entry), type 1.

  Either or both are acceptable. Key 3001 is used if both are supplied. When both key 1002 and key 3001 are added, only key 3001 is available through the command interface, but both key 1002 and key 3001 can be retrieved through APIs. When the problem type is machine detected, key 1002 is required.

- Key 5001 (contact information)

  Required if key 1 (problem log ID) is not *LOCAL. Used to enter contact data for the remote system.

- Key 6001 (history information)

  This information defines the type of create action. One or more history entries are allowed. At least one event is required.

In addition, the following information is needed based on the problem type.

Keys for Creating Problem Type 1

Machine detected problems (problem type 1) use the following keys:

Data for OPENED status are:

- Key 1002 (problem description message)

  Required.

- Key 1003 (problem creation data)

  Required.

- Key 1005 (failing device)

  Required to define the device and / or code that is failing.

- Keys 2001-2009 (FRU entries)

  At least one key of the range 2001 to 2009 is required. The keys may be provided in any order but will be stored in probability of fix order, with the highest probability FRU record being stored first and the least probability FRU record being stored last.

- Keys 4001 and 4002 (supporting data)

Optional. Used to define the supporting data that will be associated with the problem. Multiple entries are allowed.

- Data for READY status are:

Post analysis data must be added to a machine detected problem to achieve READY status. This data is in addition to the data added to achieve OPENED status. Data for READY status are:

- Key 1004 (reporting device)

  Defines the machine that will be reported to a service provider as the failing machine.
- Key 1005 (failing device) This provides a description of the resource that caused the problem.
- Key 1006 (reporting code)

  Defines the program/product that is failing. This is required if the highest probability FRU is key 2002 (code FRU type).
- Key 1007 (problem analysis data)

  Required
- Key 1010 (symptom string)

  Required
- Keys 2001-2009 (FRU entries)

  At least one key of the range 2001 to 2009 is required. The keys may be provided in any order but will be stored in probability of fix order, with the highest probability FRU record being stored first and the least probability FRU record being stored last.

Keys for Creating Problem Type 2

User-perceived hardware or software problems (problem type 2) can be created in READY and READY - PREPARED status only. These problems require data similar to machine-detected problems (problem type 1) with the following exceptions:

- Key group 2000 (FRU records) is prohibited.
- Key group 3000 (problem type 1) is required if 1002 is not used.

Keys for Creating Problem Type 3

PTF orders (problem type 3) are created in READY and READY - PREPARED status only and use the following:

- Key group 1000 (problem description entries), except key 1002 (problem description message), is prohibited.
- Key group 2000 (FRU records) is prohibited.
- Key group 4000 (supported data records) is prohibited.
- Key 1011 is optional.
- Key group 7001 is required.

Keys for Creating Problem Type 4

User-perceived remote hardware or software problems (problem type 4) can be created in READY and READY - PREPARED status only. These problems require data similar to machine-detected problems (problem type 1) with the following exceptions:

- Key group 2000 (FRU records) is prohibited.
- Key group 4000 (supported data records) is prohibited.

Keys for Creating Problem Type 5

Application detected problems (problem type 5) are used to enter problems automatically detected during the execution of a program. They can be created in READY and PREPARED status only. These problems require data similar to machine-detected problems (problem type 1) with the following exceptions:

- Key group 2000 (FRU records) is optional.

  If key group 2000 (FRU records) is specified only key 2002 (code FRU type) is permitted.

- Key group 4000 (supported data records) is optional.

  Key group 4000 (supported data records) may be used to identify APAR data that is associated with the problem.

Keys for Creating Problem Types 6 and 7

Client machine-detected (problem type 6) and user-created (problem type 7) problems are used to create problem log entries for an attached client. These problems are generated in the READY status. The data requirements are:

- Key 1012 is required and must be LOGONLY on the Add Problem Log Entry API.
- Key 1013 is required.
- Key 1003 (problem creation data)
- Key 1010
- Keys 1001, 1002, 1003
- Keys 4001 and 4002 are optional
- Key 3001 (text entry) is optional
- Key 6001 (history information) is optional

Keys for Creating Problem Type 8

User-created general problems (problem type 8) are used to add data of a general nature, that is not applicable to the types already defined. The entry can be created in READY and READY - PREPARED status only. The data requirements are:

- Key group 1000 (problem description entries) is prohibited.
- Key 1002 (problem description message) can be used to create text for the problem description of the message.
- Key 3001 (text entry) type 2 is required to provide a detailed description of the problem.

  Key group 4000 (supported data records) can be used to identify data that is associated with the problem.

Data for PREPARED Status

The supplemental data needed to move a problem from OPENED or READY status to PREPARED status are:

- Key 6 (operational data)
- Key control
- Prepared for system

  Required to define the system that this problem will be sent to.
- Key 1001 (problem severity)
- Optional. Default is none.
- Ignored for PTF Order (problem type 3).
- Key 1011 (PTF media selection)

  Optional. Default is the contact data base. Typically this is the tape device type and model or a description of the media type on which PTFs are delivered if the distribution size exceeds a predefined transmit size limit.
- Key 5001 (contact information)

  Optional. Used to override local service contact information.

Error Messages

Message ID	**Error Message Text**
CPF3C1E E	Required parameter &1 omitted.
CPF3C90 E	Literal value cannot be changed.

Message ID	Error Message Text
CPF3CF1 E	Error code parameter not valid.
CPF3CF2 E	Error(s) occurred during running of &1 API.
CPF7AAB E	Problem &1 not found.
CPF3C4D D	Length &1 for key &2 not valid.
CPF3C82 D	Key &1 not valid for API &2.
CPF3C86 D	Required key &1 not specified.
CPD7A82 D	Value not valid for key &1. (char string)
CPD7A83 D	Value not valid for key &1. (integer)
CPD7A87 D	Key &1 may be added only once.
CPD7A88 D	Incorrect DBCS field format found.
CPD7A8A D	Key value &1 is not valid.
CPD7A8B D	Length of data not valid.
CPF7A89 E	Incorrect handle for this activation.
CPF7A8A E	Problem log services not started.
CPF7AA7 E	Problem &1 not found.
CPF9821 E	Not authorized to program &1 in library &2.
CPF9872 E	Program or service program &1 in library &2 ended. Reason code &3.
CPFA320 E	Pointer parameter is null.

API introduced: V3R1

Top | "Problem Management APIs," on page 1
APIs by category

Delete Problem Log Entry (QsxDeleteProblemLogEntry) API

Required Parameter Group:

1	Handle
Input	Binary(4)
2	Key structures
Input	Array of Pointers
3	Number of keys
Input	Binary(4)
4	Error code
I/O	Char(*)

Default Public Authority: *USE

Service Program: QSXSRVPL

Threadsafe: No

The Delete Problem Log Entry (QsxDeleteProblemLogEntry) API provides one of the following functions:

- Deletes a single problem log entry.
  The problem log entry and all associated data is deleted.
- Removes data from a problem log entry.

The data that can be removed by the API are:

- Key group 2000 (FRU entries). Either all FRU entries or all FRU entries of a class are removed. Individual FRU entries cannot be removed.
- Key group 4000 (supported data entries). Either all supporting data entries are removed or all entries of a type.
- PTF entries can be deleted individually, using key 7001 (PTF ID), or they can be deleted entirely using key 7000 (PTF entry).

Authorities and Locks

API Public Authority
*USE

Required Parameter Group

Handle
INPUT; BINARY(4)

An identifier that associates the problem log services started with the QsxStartProblemLogServices API.

Key structures
INPUT; Array of Pointers

Array of pointers to the data contained in each key being passed to the API.

Number of keys
INPUT; BINARY(4)

Tells the API how many keys are being passed 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 Key Groups

Depending on the type of problem entry being deleted, one or more key groups must be provided to remove the data required or desired to the problem. You must select the keys applicable to the problem data you are deleting. If the keys provided to the API do not match the requirements for the problem log entry type you are deleting, you are notified by the error handling procedures.

For details about the keys that may be used see “Key Groups for Problem Log APIs” on page 57.

Rules for Key Usage

The Delete Problem Log Entry API can be used to:

- Delete the problem log entry.
- To delete specific entries from the problem log.

  The specific data that can be deleted are:
  - Key 2000 (class of FRU entries) to remove all FRU entries or all FRU entries of a class
  - Key 7000 (PTF entry) to remove all PTF entries
  - Key group 4000 (supporting data entries) to remove all supporting data entries, or all spooled file entries or all data file records as a group. Individual entries cannot be removed.
  - Key 7001 (PTF ID) to remove a single PTF entry

Deleting the above in any combination is supported.

The status of a problem log is not changed as a result of the delete operation.

Delete a Problem Log Entry

To delete a problem log entry, provide a key 1 (problem log ID) with no other keys. This deletes the problem log entry and all associated data. To delete the problem, it must be in CLOSED status or be older than the period defined by system value QPRBHLDITV.

Delete FRU Entries

Individual FRU entries cannot be deleted. FRU entries are deleted by class. For example, to delete the point of failure FRUs, use key 2000 and set the class field to 1. All point of failure FRUs are deleted.

Delete PTF Entries

PTF entries can be deleted individually, using key 7001 (PTF ID), or they can be deleted entirely using key 7000 (PTF entry).

Delete Supporting Data

Individual supporting data entries cannot be deleted. The entry in the problem log is deleted and the data defined by the entry is also deleted. For example, if a spooled file entry is defined, the problem log contains the name of the object to be deleted. The spooled file and the problem log entry containing the spooled file name are both deleted. Provide the following data:

- Key 1 (problem log ID)
- Key group 4000 (supported data entries) deletes all entries or all entries of a type.

Error Messages

Message ID	Error Message Text
CPF3C1E E	Required parameter &1 omitted.
CPF3C90 E	Literal value cannot be changed.
CPF3CF1 E	Error code parameter not valid.
CPF3CF2 E	Error(s) occurred during running of &1 API.
CPF7AAB E	Problem &1 not found.
CPF3C4D D	Length &1 for key &2 not valid.
CPF3C82 D	Key &1 not valid for API &2.
CPF3C86 D	Required key &1 not specified.
CPF7AA6 D	Problem record &1 cannot be deleted.
CPD7A82 D	Value not valid for key &1. (char string)
CPD7A87 D	Key &1 may be added only once.
CPD7A89 D	Record &1 was not deleted.
CPD7A8A D	Key value &1 is not valid.
CPF7A89 E	Incorrect handle for this activation.
CPF7A8A E	Problem log services not started.
CPF7AA6 E	Problem record &1 cannot be deleted.
CPF7AA7 E	Problem &1 not found.
CPF9821 E	Not authorized to program &1 in library &2.
CPF9872 E	Program or service program &1 in library &2 ended. Reason code &3.
CPFA320 E	Pointer parameter is null.

API introduced: V3R1

Top | "Problem Management APIs," on page 1
APIs by category

End Problem Log Services (QsxEndProblemLogServices) API

Required Parameter

1	Handle	
Input	Binary(4)	
2	Error code	
I/O	Char(*)	

Default Public Authority: *USE

Service Program: QSXSRVPL

Threadsafe: No

The End Problem Log Services (QsxEndProblemLogServices) API ends an instance of the problem log services identified by the handle returned when the services were started. The following are performed:

- A rollback is issued to delete any problem log entries that were not committed. This is just performed as a precaution only. The Add, Change, Create, and Delete Problem Log Entry APIs perform a commit or rollback.

Authorities and Locks

API Public Authority
*USE

Required Parameter

Handle
INPUT; BINARY(4)

The handle that defines the instance of problem log services to end.

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**
CPF3C1E E	Required parameter &1 omitted.
CPF3C90 E	Literal value cannot be changed.
CPF3CF1 E	Error code parameter not valid.
CPF7A89 E	Incorrect handle for this activation.
CPF9821 E	Not authorized to program &1 in library &2.
CPF9872 E	Program or service program &1 in library &2 ended. Reason code &3.
CPFA320 E	Pointer parameter is null.

API introduced: V3R1

Top | "Problem Management APIs," on page 1
APIs by category

Filter Problem (QSXFTRPB) API

Required Parameter Group:

1	Problem log identifier	
Input	Char(30)	
2	Error code	
I/O	Char(*)	

Default Public Authority: *USE

Threadsafe: No

The Filter Problem (QSXFTRPB) API applies the currently active problem log filter to a problem log entry.

The system value for the problem filter (QPRBFTR) identifies the active filter currently being used. Multiple filters can be defined, but only one can be active at a time. The QSXFTRPB API can be used at any time.

Required Parameter Group

Problem log identifier
INPUT; CHAR(30)

The problem to be retrieved, updated, and sent through the active filter. The problem log identifier has two parts: a problem ID number and the origin system. See "Format for the Problem Log Identifier."

Error code
I/O; CHAR(*)

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

Authorities and Locks

API Public Authority
*USE

Format for the Problem Log Identifier

Offset			
Dec	**Hex**	**Type**	**Field**
0	0	CHAR(10)	Problem ID number
10	A	CHAR(20)	Origin system

Field Descriptions

Origin system. The node name of the origin system (the format is *network ID.control point name*).

Problem ID number. The number the system generates to identify a problem.

Error Messages

Message ID	Error Message Text
CPF3C90 E	Literal value cannot be changed.
CPF3CF1 E	Error code parameter not valid.
CPF7AA7 E	Problem &1 not found.
CPF7A82 E	Error occurred while applying the problem filter.
CPF7A83 E	Problem filter &1/&2 not found.
CPF7A93 E	Problem &2 currently in use by job &1.
CPF8160 E	&8 damage on &4 type &3.
CPF9803 E	Cannot allocate object &2 in library &3.
CPF9872 E	Program or service program &1 in library &2 ended. Reason code &3.

API introduced: V2R3

Top | “Problem Management APIs,” on page 1
APIs by category

Log Software Error (QPDLOGER) API

Required Parameter Group:

1	Suspected program name	
Input	Char(10)	
2	Detection ID	
Input	Char(12)	
3	Message reference key	
Input	Char(4)	
4	Point of failure	
Input	Binary(4)	
5	Print job log	
Input	Char(1)	
6	Data items	
Input	Char(*)	
7	Data item offset and length	
Input	Array of Char(*)	
8	Number of data items	
Input	Binary(4)	
9	Object name	
Input	Array of Char(*)	
10	Number of object names	
Input	Binary(4)	
11	Error code	
I/O	Char(*)	

Optional Parameter:

12	ILE module name
Input	Char(10)

Default Public Authority: *USE

Threadsafe: Conditional; see "Usage Notes" on page 28.

The Log Software Error (QPDLOGER) API allows a program to report a software problem to the local iSeries server and provide the data needed to resolve the problem. When this API is called, any error data provided is spooled to one or more spooled files, a symptom string is created, an entry is created in the problem log, and a message is sent to the QSYSOPR message queue indicating that a software error has been detected.

Error data can be provided on the API call by using the data item offset and length and object name parameters.

Authorities and Locks

API Public Authority
*USE

Required Parameter Group

Suspected program name
INPUT; CHAR(10)

The name of the program in which the error is suspected. Service programs are not supported. The Report Software Error (QpdReportSoftwareError) API must be used to report a problem against a service program. If a service program is specified on the QPDLOGER API, the service program will not be found and the suspected program will be used instead.

Valid values are:

**SAME*	The reporting program.
**PRV*	The program that called the reporting program.
program name	The name of the suspected program.

The suspected program name is included in the symptom string (as F/*name*) created when this API is called.

Detection ID
INPUT; CHAR(12)

A message ID or other value defined by the reporting program that further identifies the problem. This value is included in the symptom string (as MSG*detectionid*) created when this API is called.

Message reference key
INPUT; CHAR(4)

The message key associated with the message that is being reported (if a message is being reported). This parameter is used to verify that message CPF9999 (a function check) was not caused by a damage exception (CPF81xx). If message CPF9999 is caused by a damage exception, the problem will not be reported. This value is ignored if it does not contain a key for a CPF9999 message.

Note: The detection ID should not contain blanks. The API ignores characters after the first blank.

Point of failure
INPUT; BINARY(4)

A return code, statement number, or other value defined by the reporting program that assists in locating the problem. This value is converted to zoned decimal and included in the symptom string (as RC*nnnnnnnn*) created when this API is called.

Print job log
INPUT; CHAR(1)

Whether the job log and other job information is to be spooled to a spooled file.

Valid values are:

Y	Print the job log and job information.
N	Do not print the job log and job information.

Data items
INPUT; CHAR(*)

The data to be spooled.

Data item offset and length
INPUT; ARRAY of CHAR(*)

An array of the offsets to and lengths of the data items to be spooled to a spooled file. The array can contain up to 32 elements.

Each element has the following structure:

Data offset	BINARY(4). The offset to the data item from the start of the data.
Data length	BINARY(4). The length of this data item (must be greater than 0).

Number of data items
INPUT; BINARY(4)

The number of elements in the array of data item offsets and lengths. The number must be between 0 and 32, inclusive.

Object name
INPUT; ARRAY of CHAR(*)

An array of object names whose contents are to be spooled to a spooled file. The array can contain up to 32 elements.

Each element has the following structure:

Object name	CHAR(30). The name of the object to be spooled.
Library	CHAR(30). The library in which the object resides.

Valid values for the library name are:

**CURLIB*
The job's current library.

**LIBL* The library list.

library name
The specific library that contains the object.

Object type
CHAR(10). The object type. For a complete list of the available object types, see Object Types in the CL topic.

Number of object names
INPUT; BINARY(4)

The number of object names in the array of object names. The number must be between 0 and 32, inclusive.

Error code
I/O; CHAR(*)

The structure in which to return error information. For the format of the structure, see Error Code Parameter. If this parameter is omitted, diagnostic and escape messages are issued to the application.

Optional Parameter

ILE module name

INPUT; CHAR(10)

The name of the integrated language environment (ILE) module in which the error is suspected. This value is included in the symptom string created when this API is called.

Usage Notes

When this API runs within a threaded job, no problem log entry is created. When the API is called, the following occurs:

- Any error data that is provided is spooled to one or more spooled files.
- A symptom string is created.
- A message is sent to the job log and to the QSYSOPR message queue, which indicates that a software error has been detected.

Error data can be provided on the call to the API by using the data item offset and length parameters. (No object dumping support is available).

Also, dump job output is provided to help with problem determination.

The following parameters are ignored:

- Print job log
- Object name
- Number of object names

Current API users do not have to make any changes.

Error Messages

Message ID	Error Message Text
CPF3C90 E	Literal value cannot be changed.
CPF3CF1 E	Error code parameter not valid.
CPF93C0 E	Software error logging not active.
CPF93C2 E	&1 is not a valid number of data items.
CPF93C3 E	&1 is not a valid number of object names.
CPF93C4 E	Error already logged.
CPF93C5 E	Software problem logging (QPDLOGER) API error occurred.
CPF9872 E	Program or service program &1 in library &2 ended. Reason code &3.
CPI93B2 I	Software problem data for &4 has been detected.
CPI93CA I	Suspected program &1 not found.
CPI93CB I	Point-of-failure value not valid.
CPI93CC I	Object &1 in library &2 not found.
CPI93CF I	Data length or data offset not valid.

API introduced: V2R3

Top | “Problem Management APIs,” on page 1
APIs by category

Report Software Error (QpdReportSoftwareError) API

Required Parameter Group:

1	Problem description records	
Input	Array of Pointers	
2	Number of problem description records	
Input	Binary(4)	
3	Error code	
I/O	Char(*)	

Default Public Authority: *USE

Service Program: QPDSRVPG

Threadsafe: Conditional; see “Usage Notes” on page 38.

Use the Report Software Error (QpdReportSoftwareError) API whenever your ILE program detects a software problem that must be fixed.

The API logs the problem in the system problem log, which lets you track the problem, as well as send it to a service provider. See the System Manager for iSeries product for more information about service providers and service requesters.

The API also lets you specify the symptoms that identify the problem. The operating system and the service provider use those symptoms to find a PTF that may fix the problem.

The API also lets you specify data to dump to spooled files. If neither the operating system nor the service provider can find a PTF, you may be able to find the cause of the problem using the data the program dumped.

Authorities and Locks

Object Authority

*USE for objects in libraries

*R for objects in directories

Object Library Authority
*EXECUTE

Object Directory Authority
*RX

Locks None

Required Parameter Group

Problem description records
INPUT; ARRAY of POINTERS

This is a list of pointers to problem symptom and data description records. See “Problem Description Records Format” for the format of the records.

Number of Problem description records
INPUT; BINARY(4)

The number of problem description records your program is passing to the API.

Error code
I/O; CHAR(*)

The structure in which to return error information. For the format of the structure, see Error Code Parameter. If this parameter is omitted, diagnostic and escape messages are issued to the application.

Problem Description Records Format

Offset		Type	Field
Dec	**Hex**		
0	0	BINARY(4)	Key
4	4	CHAR(*)	Problem record description

Field Descriptions

Key Identifies a particular problem symptom or data. See “Keys” for a list of the possible keys.

Problem record description This describes a particular symptom of the problem, or specifies data to collect. See “Formats of Specific Problem Description Records” on page 31 for the various problem record description formats.

Keys

The following table lists the valid keys of the key field area of the software problem record.

Key	Description
100	“Key 100-Call Stack Counter” on page 31
101	“Key 101-Suspected Program” on page 31
102	“Key 102-Suspected Service Program” on page 31
103	“Key 103-Suspected Module” on page 32
104	“Key 104-Suspected Procedure” on page 32
105	“Key 105-Detecting Program” on page 32
106	“Key 106-Detecting Service Program” on page 33
200	“Key 200-Symptom” on page 33
201	“Key 201-Instruction Number” on page 33
300	“Key 300-System Object” on page 34
301	“Key 301-Data” on page 34
302	“Key 302-Named System Object” on page 34
303	“Key 303-Spooled File” on page 35
304	“Key 304-Named Integrated File System Object” on page 35
400	“Key 400-Service Identifier” on page 35

Formats of Specific Problem Description Records

Key 100-Call Stack Counter

This key specifies which invocation on the program stack is suspected of causing the error being reported. If this key is used, you must not use keys 101, 102, 103, or 104. If neither key 100, 101, nor 102 are specified, the API assumes that the program or service program that called the API is the one that has the problem.

Offset			
Dec	**Hex**	**Type**	**Field**
0	0	BINARY(4)	Key
4	4	BINARY(4)	Call stack counter

Key 101-Suspected Program

This key is used to identify which program is suspected of causing the error being reported. If this key is used, you must not use key 100 or 102, but should use keys 103 and 104 if applicable. If neither key 100, 101, nor 102 are specified, the API assumes that the program or service program that called the API is the one that has the problem.

Note: The program must exist on the system at the time the API is called.

Offset			
Dec	**Hex**	**Type**	**Field**
0	0	BINARY(4)	Key
4	4	BINARY(4)	Length of program name
8	8	BINARY(4)	Length of library name
12	C	CHAR(4)	Reserved
16	10	POINTER	Program name
32	20	POINTER	Library name

Key 102-Suspected Service Program

This key is used to identify which service program is suspected of causing the error being reported. If this key is used, you must not use key 100 or 101, but should use keys 103 and 104 if applicable. If neither key 100, 101, nor 102 are specified, the API assumes that the program or service program that called the API is the one that has the problem.

Note: The service program must exist on the system at the time the API is called.

Offset			
Dec	**Hex**	**Type**	**Field**
0	0	BINARY(4)	Key
4	4	BINARY(4)	Length of service program name
8	8	BINARY(4)	Length of library name
12	C	CHAR(4)	Reserved

Offset			
Dec	**Hex**	**Type**	**Field**
16	10	POINTER	Service program name
32	20	POINTER	Library name

Key 103-Suspected Module

This key is used to identify which module is suspected of causing the error being reported. If this key is used, you must not use key 100, but should use keys 101 or 102.

Note: The module must exist on the system at the time the API is called.

Offset			
Dec	**Hex**	**Type**	**Field**
0	0	BINARY(4)	Key
4	4	BINARY(4)	Length of module name
8	8	BINARY(4)	Length of library name
12	C	CHAR(4)	Reserved
16	10	POINTER	Module name
32	20	POINTER	Library name

Key 104-Suspected Procedure

This key is used to identify which procedure is suspected of causing the error being reported. If this key is used, you must not use key 100, but should use key 103 and either 101 or 102.

Offset			
Dec	**Hex**	**Type**	**Field**
0	0	BINARY(4)	Key
4	4	BINARY(4)	Length of procedure name
8	8	CHAR(8)	Reserved
16	10	POINTER	Procedure name

Key 105-Detecting Program

This key identifies the program that detected the problem. If this key is used, you must not use key 106. If neither key 105 nor 106 is specified, the API assumes that the program or service program that called the API is the one that detected the problem.

Note: The program must exist on the system at the time the API is called.

Offset			
Dec	**Hex**	**Type**	**Field**
0	0	BINARY(4)	Key
4	4	BINARY(4)	Length of program name
8	8	BINARY(4)	Length of library name
12	C	CHAR(4)	Reserved

Offset		Type	Field
Dec	Hex		
16	10	POINTER	Program name
32	20	POINTER	Library name

Key 106-Detecting Service Program

This key identifies the service program that detected the problem. If this key is used, you must not use key 105. If neither key 105 nor 106 is specified, the API assumes that the program or service program that called the API is the one that detected the problem.

Note: The service program must exist on the system at the time the API is called.

Offset		Type	Field
Dec	Hex		
0	0	BINARY(4)	Key
4	4	BINARY(4)	Length of service program name
8	8	BINARY(4)	Length of library name
12	C	CHAR(4)	Reserved
16	10	POINTER	Service program name
32	20	POINTER	Library name

Key 200-Symptom

This key identifies the symptoms associated with the problem. Together, the symptoms form a symptom string. OS/400 searches for a PTF that has a solution string that matches this symptom string.

Offset		Type	Field
Dec	Hex		
0	0	BINARY(4)	Key
4	4	BINARY(4)	Length of symptom keyword
8	8	BINARY(4)	Length of symptom data
12	C	CHAR(1)	Type of symptom data
13	D	CHAR(3)	Reserved
16	10	POINTER	Pointer to symptom keyword
32	20	POINTER	Pointer to symptom data

Key 201-Instruction Number

This key identifies the instruction number where the problem occurred.

Offset		Type	Field
Dec	Hex		
0	0	BINARY(4)	Key
4	4	CHAR(4)	Instruction number

Key 300-System Object

This key identifies system objects associated with the problem. The system objects will be dumped to spooled files. The spooled files will be kept on an output queue in the APAR library associated with the problem log entry. You can display the spooled files using the WRKPRB command. The combination of this key and the other keys related to objects may be specified up to 32 times.

Offset			
Dec	**Hex**	**Type**	**Field**
0	0	BINARY(4)	Key
4	4	CHAR(12)	Reserved
16	10	PTR(SYP)	Pointer to object

Key 301-Data

This key identifies data associated with the problem. The data is dumped to spooled files. This key may be specified up to 32 times. The spooled files are kept on an output queue in the APAR library associated with the problem log entry. You can display the spooled files using the WRKPRB command. The first one thousand bytes from the list of data items are also sent to the service provider if the problem is reported and if the “send data packet” flag in the service attributes is on. That data resides in a file named QAPDFCDP in the APAR library associated with the problem log entry on the service provider.

Offset			
Dec	**Hex**	**Type**	**Field**
0	0	BINARY(4)	Key
4	4	BINARY(4)	Data length
8	8	BINARY(4)	Data ID
12	C	CHAR(4)	Reserved
16	10	POINTER	Pointer to data

Key 302-Named System Object

This key names system objects associated with the problem. The system objects will be dumped to spooled files. The spooled files will be kept on an output queue in the APAR library associated with the problem log entry. You can display the spooled files using the WRKPRB command. The combination of this key and the other keys related to objects may be specified up to 32 times.

Note: The object must exist on the system at the time the API is called.

Offset			
Dec	**Hex**	**Type**	**Field**
0	0	BINARY(4)	Key
4	4	CHAR(30)	Object name
34	22	CHAR(30)	Object library
64	40	CHAR(10)	Object type

Key 303-Spooled File

This key identifies spooled files associated with the problem. The job that created the spooled files must be the current job. This key may be specified up to 32 times. The spooled files are kept on an output queue in the APAR library associated with the problem log entry.

Offset			
Dec	**Hex**	**Type**	**Field**
0	0	BINARY(4)	Key
4	4	CHAR(10)	Spooled file name
14	E	BINARY(4)	Spooled file number

Key 304-Named Integrated File System Object

This key names integrated file system objects associated with the problem. The integrated file system objects will be dumped to spooled files. The spooled files will be kept on an output queue in the APAR library associated with the problem log entry. You can display the spooled files using the WRKPRB command. The combination of this key and the other keys related to objects may be specified up to 32 times.

Notes:

1. The object must exist on the system at the time the API is called.
2. Both absolute and relative path names are allowed. The patterns ? and * are not allowed. The home directory of the user is not resolved, thus a tilde (~) in the first character position is not treated as the home directory. The NLS-enabled path name structure (defined in the QLG header file) can be filled in to specify the coded character set identifier (CCSID) the path name is in.

Offset			
Dec	**Hex**	**Type**	**Field**
0	0	BINARY(4)	Key
4	4	CHAR(12)	Reserved
16	10	POINTER	NLS-enabled path name structure

Key 400-Service Identifier

This key identifies where in a particular program or service program the problem was reported. The default service identifier is 9000.

Offset			
Dec	**Hex**	**Type**	**Field**
0	0	BINARY(4)	Key
4	4	CHAR(4)	Service identifier

Field Descriptions

Call stack counter. The number of invocations in the program stack to count from the invocation of the program or service program that called the API, to the invocation of the program or service program that is suspected of having the problem. Use 1, for instance, to specify the program or service program that called the program or service program that called the API. If the call stack counter value exceeds the number of invocations currently on the program stack, the API uses the invocation of the program or service program that called the API.

Data ID. This number is used to identify the data that is dumped.

Data length. The length of the data that is dumped.

Instruction number. Specifies exactly where the problem occurred within the specified program or service program.

Key. Identifies the problem description record.

Length of library name. The length of the library name. The value ranges from 1 to 10.

Length of module name. The length of the module name. The value ranges from 1 to 10.

Length of procedure name. The length of the procedure name. The value ranges from 1 to 256.

Length of program name. The length of the program name. The value ranges from 1 to 10.

Length of service program name. The length of the service program name. The value ranges from 1 to 10.

Length of symptom data. This indicates how many bytes the stored data occupies. The valid range is 1 to 15. The length of the symptom data plus the length of the symptom keyword must not exceed 15.

Length of symptom keyword. The length of the symptom keyword. The valid range is 1 to 15. The length of the symptom data plus the length of the symptom keyword must not exceed 15.

Library name. A pointer to the name of the library which contains the program, service program, or module in which the error has occurred.

Module name. A pointer to the name of the module in which the error has occurred.

NLS-enabled path name structure. For more information on this structure, see Path Name Format.

Object library. The library in which the object resides.

Valid values for the library name are:

**CURLIB*	The job's current library.
**LIBL*	The library list.
library name	The specific library that contains the object.

Object name. The name of the object to be dumped.

Object type. The type of object. For complete list of the available object types, see the Control Language (CL) information in the iSeries Information Center.

Pointer to data. A space pointer to the data.

Pointer to object. A system pointer to a system object.

Pointer to symptom data. A pointer to the symptom data. The symptom data is a symptom of the problem. It is concatenated to the symptom keyword. The sum of the symptom keyword length and the symptom data length must not be longer than 15 characters.

Pointer to symptom keyword. A pointer to the system keyword. The symptom keyword is concatenated to the symptom data. The sum of the symptom keyword length and the symptom data length must not be longer than 15 characters. There are a limited number of keywords that can be used. The valid keywords are:

Table 1. Symptom string keywords

Key	Description
(blanks)	Normally, a symptom in the symptom string consists of a keyword and data. However, for flexibility, you may specify a symptom without a keyword.
MSG	This is a message identifier associated with the problem.
RC	The point of failure is a number that identifies a subroutine, block of code, or specific statement associated with the problem. Note: This number should not be an instruction number, since the instruction number may be different for different versions of the same program.
FLDS/	This is the name of a field associated with the problem. It may be followed by the VALU/ keyword to show what value the field contained at the time of the failure.
MOD/	MOD/ is the name of the ILE module that might have caused the problem being reported.
OPCS/	OPCS/ is the name of the command, macro, or instruction associated with the problem.
PCSS/	PCSS/ is a program label that shows generally where the problem occurred.
PRCS/	PRCS/ is a reason code or return code associated with the problem.
REGS/	This is the name of a register associated with the problem. It may be followed by the VALU/ keyword to show what value the register contained at the time of the failure.
RIDS/	RIDS/ is the name of the subroutine or the identifier of the thread in which the problem occurred.
VALU/	VALU/ is the value of a field or register at the time the problem occurred. VALU/ must appear after key FLDS/ or REGS/.

Procedure name. A space pointer to the name of the procedure in which the error has occurred.

Program name. A pointer to the name of the program in which the error is suspected. The suspected program name is included in the symptom string (as F/name) created when this API is called. If neither the 100, 101, nor the 102 keys are used, then the program name in the symptom string defaults to the caller of this API.

Reserved. Null.

Service identifier. Identifiers where in a program or service program the problem was reported. The valid range is 1 to 8999.

Service program name. A pointer to the name of the service program in which the error is suspected. The suspected service program name is included in the symptom string (as F/name) created when this API is called. If the 100, 101, or the 102 keys are not used, then the service program name in the symptom string defaults to the caller of this API.

Spooled file name. The name of a spooled file associated with the problem.

Spooled file number. The unique number of a spooled file associated with the problem. The valid range is 1 through 9999.

The following special values are supported for this parameter:

0	Only one spooled file from the job has the specified file name, so the number of the spooled file is not necessary.
-1	This uses the highest numbered spooled file with the specified file name.

Type of symptom data. This indicates how the data is stored.

The possible values are:

C	The data is in displayable form. It must not include blanks or characters that are not displayable.
X	The data is in hexadecimal form. The API converts it to displayable characters. **Note:**The length of symptom data is the number of bytes used to store the hexadecimal value.
D	The data is in zoned decimal form.
P	The data is in packed decimal form. The API converts it to displayable numbers.
B	The data is in binary form. The API converts it to displayable numbers. **Note:**The length of symptom data can only be 2 or 4 bytes if the type of symptom data is B.

Usage Notes

When this API runs within a threaded job, no problem log entry is created. When the API is called, the following occurs:

- Any error data that is provided is spooled to one or more spooled files.
- A symptom string is created.
- A message is sent to the job log and to the QSYSOPR message queue, which indicates that a software error has been detected.

Error data can be provided on the call to the API by using the data item offset and length parameters. (No object dumping support is available).

Also, dump job output is provided to help with problem determination.

Also, the following keys are ignored:

Key	Description
300	System object
302	Named system object
303	Spooled file
400	Service identifier

Error Messages

Message ID	**Error Message Text**
CPF3C90 E	Literal value cannot be changed.
CPF3CF1 E	Error code parameter not valid.
CPF3CF2 E	Error(s) occurred during running of &1 API.
CPF3C82 D	Key &1 not valid for API &2.
CPF3C85 D	Value for key &1 not allowed with value for key &2.
CPF93C2 D	&1 is not a valid number of data items.
CPF93C3 D	&1 is not a valid number of object names.
CPF93C8 D	Not a valid number of symptoms.
CPF93C9 D	Not a valid number of spooled files.

Message ID	Error Message Text
CPF93C0 E	Software error logging not active.
CPF93C4 E	Error already logged.
CPF93C6 E	Suspected program cannot be determined.
CPF93C7E	Error in parameter &1.
CPF9872 E	Program or service program &1 in library &2 ended. Reason code &3.
CPI93B2 I	Software problem data for &4 has been detected.

API introduced: V3R1

Top | "Problem Management APIs," on page 1
APIs by category

Retrieve Policy Data (QPDETRTV) API

Required Parameter Group:

1	Receiver variable	Output	Char(*)
2	Length of receiver variable	Input	Binary(4)
3	Format name	Input	Char(8)
4	Error Code	I/O	Char(*)

Default Public Authority: *EXCLUDE

Threadsafe: Yes

The Retrieve Policy Data (QPDETRTV) API retrieves policy data.

Authorities and Locks

None.

Required Parameter Group

Receiver variable
OUTPUT; CHAR(*)

The variable that will receive the policy information being retrieved. For the format, see "Format of Data Returned" on page 40.

Length of receiver variable
INPUT; BINARY(4)

The length of the receiver variable described in Format of data returned. If the length is larger than the size of the receiver variable, the results may not be predictable. The minimum length is 8 bytes.

Format name
INPUT; CHAR(8)

The format of the information to be returned. You must use one of the following format names:

RPOL0300
Retrieve maximum PTF order size.

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 Data Returned

The receiver variable holds the policy information returned.

RPOL0300 - Retrieve maximum PTF order size

Offset			
Dec	**Hex**	**Type**	**Field**
0	0	Binary(4)	Bytes returned
4	4	Binary(4)	Bytes available
8	8	Binary(4)	Maximum PTF order size over LAN
12	C	Binary(4)	Maximum PTF order size over a modem

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.

Maximum PTF order size over a modem. The maximum size in megabytes for a PTF order to be delivered over a modem. A value of -1 indicates PTF orders of any size are delivered over a modem. A value of 100 MB (MB equals approximately 1 000 000 bytes) is used if a lower value is retrieved.

Maximum PTF order size over LAN. The maximum size in megabytes for a PTF order to be delivered over the local area network (LAN). A value of -1 indicates PTF orders of any size are delivered over the LAN. A value of -1 is used if a value lower than 100 MB (MB equals approximately 1 000 000 bytes) is retrieved.

Error Messages

The following messages may be sent from this function:

Message ID	**Error Message Text**
CPF3CF1	Error code parameter not valid.
CPF3CF2	Error(s) occurred during running of * API.
CPF3C1D	Input variable length in parameter * not valid.
CPF3C1E	Required parameter * omitted.
CPF3C17	Error occurred with input data parameter.
CPF3C21	Format name * is not valid.
CPF3C4A	Value not valid for field *.
CPF3C4B	Value not valid for field *.
CPF3C4C	Value not valid for field *.
CPF9872	Program or service program * in library * ended. Reason code *.

API introduced: V5R3 with PTF SI20372

Top | "Problem Management APIs," on page 1 | APIs by category

Retrieve Problem Log Entry (QsxRetrieveProblemLogEntry) API

Required Parameter Group:

1	Handle
Input	Binary(4)
2	Key structures
Input	Array of Pointers
3	Number of keys
Input	Binary(4)
4	Receiver variable
Output	Char(*)
5	Length of receiver variable
Input	Binary(4)
6	Return information
Output	Char(16)
7	Pointer to array of pointers to the keys returned in the buffer receiver variable
Output	PTR(SPP)
8	Error code
I/O	Char(*)

Default Public Authority: *USE

Service Program: QSXSRVPL

Threadsafe: No

The Retrieve Problem Log Entry (QsxRetrieveProblemLogEntry) API allows a user to extract data from a specific problem log entry, which the caller identifies. The problem log entry is identified by key 1 (problem log ID). The data to be retrieved is identified by the keys passed by reference. The keys used to identify the data to be retrieved are not changed by the API.

Data is returned in the receiver variable. If you are supplying an automatically extendable space, specify -1 for the size of the receiver variable. You provide the size and location of this receiver variable. If the receiver variable is not large enough to contain all the keys requested, the keys successfully retrieved to that point are returned. The number of keys returned is set in the return information parameter (number 6).

The API can be used to:
- Read a specific key from a problem log entry
- Read a group of keys

Authorities and Locks

QSXSRVPL authority
: *USE

API Public Authority
*USE

Required Parameter Group

Handle
INPUT; BINARY(4)

An identifier that associates the problem log services started with the QsxStartProblemLogServices API.

Key structures
INPUT; ARRAY of POINTERS

List of keys defining the data to be returned.

Number of keys
INPUT; BINARY(4)

Number of keys passed to the API in the input key array.

Receiver variable
OUTPUT; CHAR(*)

The variable that provides the output buffer.

Length of receiver variable
INPUT; BINARY(4)

The size of the output buffer. If it is -1, an automatically extendable space is assumed.

Return information
OUTPUT; CHAR(16)

- Bytes returned—BINARY(4)
- Bytes available—BINARY(4)
- Number of keys returned—BINARY(4)
- Reserved—BINARY(4)

Pointer to array of pointers to the keys returned in the buffer receiver variable
OUTPUT; PTR(SPP)

The pointer to the array of pointers to the keys returned in the buffer.

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 Key Groups

For details about the keys that can be used, see “Key Groups for Problem Log APIs” on page 57

Rules for Key Usage

Any amount and type of data can be retrieved from the problem log. The limiting factor is the size of the buffer that is available. The data is returned in a buffer up to the size of the buffer.

Data to be retrieved must be identified by the keys provided. Key 1 (problem log ID) is required. All other keys are optional, but only data for valid keys defined is returned.

Data, including PTF entries, can be returned individually or in groups. Data that will be returned as a group are:

- FRU entries
- Text entries
- Supporting data entries
- History entries

Retrieve PTF records

PTF data can be retrieved individually or as a group.

To retrieve all PTFs, use key 7000 (PTF entry). Key 7000 is returned and the count field states how many Key 7001 (PTF ID) keys are returned. For key 7001, the data required includes PTF ID, product ID, version, release, and modification level.

To retrieve individual PTFs, use key 7001 (PTF ID) and add the fields that are to be used as the key. Key 1 (problem log ID) and key 7001 (PTF ID) are required. Product data is optional, but can be required if multiple PTFs have the same PTF identifier.

Retrieve FRU records

To retrieve a FRU group, provide key 2000 (class of FRU entries) and a class. Key 2000 (class of FRU entries) is returned with a count of FRU entries and 2000—2009 are returned.

Retrieve text records

To retrieve the text data, provide key 3000 (text entry) and the text type. Key 3000 (text entry) is returned with a count of key 3001 (text entry) returned. Only one is returned unless all text was requested.

Retrieve supporting data

To retrieve supporting data records, provide key group 4000 (supporting data entries) and the type. Key group 4000 is returned with a count of the entries and key 4001 (spooled file data) and 4002 are returned.

Retrieve history records

To retrieve the history data, provide key 6000 (history information) and specify last or all. Key 6000 (history information) is returned with a count of the history entries and key 6001 (history information) is returned.

Retrieve analyzed error flag:

To retrieve the analyzed eror flag data, provide key 8000 (analyzed error flag entries). Key 8000 returns a value that indicates whether the problem has been analyzed by System Licensed Internal Code (SLIC).

Retrieve logical partition ID:

To retrieve the logical partition ID data, provide key 9000 (logical partition ID). Key 9000 returns the current logical partition ID on the physical machine.

Error Messages

Message ID	**Error Message Text**
CPF3C1E E	Required parameter &1 omitted.
CPF3C90 E	Literal value cannot be changed.
CPF3CF1 E	Error code parameter not valid.
CPF3CF2 E	Error(s) occurred during running of &1 API.
CPF7AAB E	Problem &1 not found.
CPF3C4D D	Length &1 for key &2 not valid.
CPF3C82 D	Key &1 not valid for API &2.

Message ID	Error Message Text
CPF3C86 D	Required key &1 not specified.
CPD7A82 D	Value not valid for key &1. (char string)
CPD7A84 D	Buffer area not accessible.
CPD7A87 D	Key &1 may be added only once.
CPD7A8A D	Key value &1 is not valid.
CPF7AA7 E	Problem &1 not found.
CPF9821 E	Not authorized to program &1 in library &2.
CPF9872 E	Program or service program &1 in library &2 ended. Reason code &3.
CPFA320 E	Pointer parameter is null.

API introduced: V3R1

Top | “Problem Management APIs,” on page 1
APIs by category

Retrieve Service Attributes (QESRSRVA) API

Required Parameter Group:

1	Receiver variable
Output	Char(*)
2	Length of receiver variable
Input	Binary(4)
3	Number of service attribute keys
Input	Binary(4)
4	Service attribute keys
Input	Array(*) of Binary(4)
5	Error code
I/O	Char(*)

Default Public Authority: *USE

Threadsafe: No

The Retrieve Service Attributes (QESRSRVA) API copies specified service attributes into the receiver variable.

Required Parameter Group

Receiver variable
OUTPUT; CHAR(*)

The variable in which this API returns the data. See “Receiver Variable Format” on page 45

Length of receiver variable
INPUT; BINARY(4)

The length of the receiver variable. The length of the receiver variable is 16 times the number of service attributes to be retrieved, plus the length of each service attribute retrieved, plus 4.

As an example, the size of the receiver variable needed to retrieve the automatic problem analysis and automatic problem reporting attributes is (16 * 2) + 1 + 1 + 4.

Note: If this value is larger than the actual size of the receiver variable, the results may not be predictable.

Number of service attribute keys
INPUT; BINARY(4)

The total number of service attributes to retrieve.

Service attribute keys
INPUT: ARRAY(*) of BINARY(4)

A list of keys that identify which service attributes to retrieve. The keys and their associated service attributes are:

Key	Service attribute
1	Automatic problem analysis
2	Automatic problem reporting
3	Service provider to report problem
4	PTF install type
5	Critical message recipients
6	Send data packets
10	System-disabled reporting connection number
11	System-disabled call-back connection number
12	Service provider connection number

Error code
I/O; CHAR(*)

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

Receiver Variable Format

The format of the receiver variable is:

Offset			
Dec	Hex	Type	Field
0	0	BINARY(4)	Number of service attributes retrieved
4	4	ARRAY(*) of BINARY(4)	Offsets to service attribute templates
*	*	CHAR(*)	Service attribute templates

Field Descriptions

Number of service attributes retrieved.

The number of service attributes the API put into the receiver variable. This number will be less than the number requested if the receiver variable is too small.

Offsets to service attribute templates. A list of values. Each value is an offset from the beginning of the receiver variable to a service attribute template.

Service attribute templates. The templates of the requested service attributes. There is one template for each service attribute retrieved. The formats of the templates are shown in "Service Attribute Template Format"

Service Attribute Template Format

The format of a service attribute template is:

Offset			
Dec	**Hex**	**Type**	**Field**
0	0	BINARY(4)	Service attribute key
4	4	BINARY(4)	Length of service attribute
8	8	CHAR(1)	Data type of service attribute
9	9	CHAR(1)	Status of service attribute
10	10	CHAR(2)	Reserved
12	C	CHAR(*)	Service attribute

Field Descriptions

Data type of service attribute. The type of data returned.

0	The service attribute was not available.
1	The service attribute is returned in character format.
2	The service attribute is returned in binary format.

Length of service attribute. The length of the service attribute. If the service attribute was not available, this value is 0.

Reserved. This field must contain null characters.

Service attribute. The requested service attribute. See "Service Attributes Format" for the formats of the service attributes.

Service attribute key. A value that identifies the service attribute that was retrieved.

Status of service attribute. Wwhether the service attribute was available for retrieval.

0	The service attribute was available.
1	The service attribute was locked.

Service Attributes Format

The Service Attributes Format has the following self-explanatory keys to solve problems:

- "Key 1—Automatic Problem Analysis" on page 47
- "Key 2—Automatic Problem Reporting" on page 47
- "Key 3—Service Provider to Report Problem" on page 47

- “Key 4—PTF Install Type” on page 48
- “Key 5—Critical Message Recipients” on page 48
- “Key 6—Send Data Packets” on page 49
- “Key 10—System-Disabled Reporting Connection Number” on page 49
- “Key 11—System-Disabled Call-Back Connection Number” on page 49
- “Key 12—Service Provider Connection Number” on page 50

Key 1—Automatic Problem Analysis

Offset			
Dec	**Hex**	**Type**	**Field**
0	0	CHAR(1)	Attribute

Field Descriptions

Attribute. The problem analysis attribute specifies when to analyze problems.

0	Problems will not be analyzed when they are logged. Instead, the operator must analyze the problem from the QSYSOPR message queue or from the Work with Problems (WRKPRB) command.
1	The system will analyze the problem as soon as the problem is logged.

Key 2—Automatic Problem Reporting

Offset			
Dec	**Hex**	**Type**	**Field**
0	0	CHAR(1)	Attribute

Field Descriptions

Attribute. The problem reporting attribute specifies when to report problems.

0	Problems will not be reported when they are logged. Instead, the operator must report the problem from the QSYSOPR message queue or from the Work with Problems (WRKPRB) command.
1	If the problem analysis attribute specifies that problems are to be analyzed as soon as the problem is logged, the system will report the problem to the service provider specified in the Service provider to report problem attribute as soon as the problem is analyzed.

Key 3—Service Provider to Report Problem

Offset			
Dec	**Hex**	**Type**	**Field**
0	0	CHAR(1)	Name format
1	1	CHAR(17)	Service provider name

Field Descriptions

Name format. This is an 'A' to show that the name is an SNA node name.

Service provider name. This identifies the service provider to report problems to if the automatic problem reporting' attribute specifies that problems are to be reported as soon as a problem is analyzed. If this field contains *IBMSRV, problems will be sent to IBM. Otherwise, the first eight characters of this field contain the control point name of the service provider. The next nine characters contain either the network identifier of the service provider, or *LCLNETID if the network identifier of the service provider is the same as that of the system that is reporting the problem.

Key 4—PTF Install Type

Offset			
Dec	**Hex**	**Type**	**Field**
0	0	CHAR(10)	Type of PTF install

Field Descriptions

Type of PTF install. This service attribute determines whether the immediate PTFs are applied immediately or delayed.

**DLYIPL*	All PTFs will be marked for delayed apply and the system will be IPLed.
**DLYALL*	All PTFs will be marked for delayed apply and the system will not be IPLed.
**IMMONLY*	The immediate PTFs will be applied and the delayed PTFs marked for apply at the next IPL.
**IMMDLY*	Only the immediate PTFs will be applied and the system will not be IPLed.

Key 5—Critical Message Recipients

Offset			
Dec	**Hex**	**Type**	**Field**
0	0	BINARY(4)	Number of entries
4	4	ARRAY(50) of CHAR(10)	User list

Field Descriptions

Number of entries. This is the number of entries in the user list.

User list. This is an ordered list of user identifiers and user classes. If the system detects a critical condition such as a DASD failure, and the first entry in this list is a user identifier, and that user is signed on, the system will send a break message to that user. If the first entry is a user class, the system will try to send a break message to all the users in that class that are signed on.

If the specified user is not signed on, or none of the users in the user class are signed on, the system tries to send the break message to the user identifier or user class in the second entry of this list.

The system keeps trying to find a user that is signed on until it reaches the end of the list.

This function is only used if problem analysis routines are run automatically at the time of failure (the ANZPRBAUTO service attribute is *YES).

**SYSOPR*	All users of user class *SYSOPR will receive a message when a critical message is sent.
**SECOFR*	All users of user class *SECOFR will receive a message when a critical message is sent.
**SECADM*	All users of user class *SECADM will receive a message when a critical message is sent.
**PGMR*	All users of user class *PGMR will receive a message when a critical message is sent.
**USER*	All users of user class *USER will receive a message when a critical message is sent.

Key 6—Send Data Packets

Offset			
Dec	**Hex**	**Type**	**Field**
0	0	CHAR(1)	Attribute

Field Descriptions

Attribute. The Send data packets attribute specifies whether or not to send problem data to the service provider.

0	Data will not be sent to the service provider.
1	Up to 2000 bytes of data will be sent to the service provider.

Key 10—System-Disabled Reporting Connection Number

Offset			
Dec	**Hex**	**Type**	**Field**
0	0	CHAR(30)	System-disabled reporting connection number

Field Descriptions

System-disabled reporting connection number. The complete electronic connection number used for automatic reporting to external support when this system is disabled. This number should include the entire sequence of numbers required to complete the call, including international access codes, country or region codes, area codes, exchanges, and so on, as appropriate.

Key 11—System-Disabled Call-Back Connection Number

Offset			
Dec	**Hex**	**Type**	**Field**
0	0	CHAR(30)	System-disabled call-back connection number

Field Descriptions

System-disabled call-back connection number. The complete electronic connection number used to call back this system from external support when this system is disabled. This number should include the entire sequence of numbers required to complete the call, including international access codes, country or region codes, area codes, exchanges, and so on, as appropriate.

Key 12—Service Provider Connection Number

Offset			
Dec	**Hex**	**Type**	**Field**
0	0	CHAR(30)	Service provider connection number

Field Descriptions

Service provider connection number. The complete electronic connection number to the service provider. This number should include the entire sequence of numbers required to complete the call, including international access codes, country or region codes, area codes, exchanges, and so on, as appropriate.

Error Messages

Message ID	Error Message Text
CPF24B4 E	Severe error while addressing parameter list.
CPF3C19 E	Error occurred with receiver variable specified.
CPF3C90 E	Literal value cannot be changed.
CPF3CF1 E	Error code parameter not valid.
CPF8C50 E	Key in input list not valid.
CPF8C51 E	Error with receiver variable length.
CPF8C52 E	Number of values in input list not valid.
CPF9872 E	Program or service program &1 in library &2 ended. Reason code &3.

API introduced: V3R1

Top | "Problem Management APIs," on page 1 | APIs by category

Set User Policy (QPDETPOL) API

Required Parameter Group:

1	Policy data	Input	Char(*)
2	Length of policy data	Input	Binary(4)
3	Format of policy data	Input	Char(8)
4	Error Code	I/O	Char(*)

Default Public Authority: *EXCLUDE

Threadsafe: Yes

The Set User Policy (QPDETPOL) API allows the changing of user policies related to service. This includes:

- What is the maximum size for a PTF order to be delivered electronically

Authorities and Locks

User Profile Authority

*SERVICE

Required Parameter Group

Policy data
INPUT; CHAR(*)

Information to use when setting the policy.

Length of policy data
INPUT; BINARY(4)

How long the policy data is.

Format of policy data
INPUT; CHAR(8)

Which policy to set. Only the following values are accepted.

"POLS0300 - Format for setting maximum PTF order size"	Set maximum PTF order size.

Error code
I/O; CHAR(*)

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

POLS0300 - Format for setting maximum PTF order size

The following information needs to be supplied in the policy data parameter (parameter 1) for the POLS0300 format.

Offset			
Dec	**Hex**	**Type**	**Field**
0	0	BINARY(4)	Maximum PTF order size over LAN
4	4	BINARY(4)	Maximum PTF order size over a modem

Field Descriptions

Maximum PTF order size over a modem. The maximum size in megabytes for a PTF order to be delivered electronically over a modem. This policy is shipped with a default value of 100 MB (MB equals approximately 1 000 000 bytes). If -1 is specified, PTF orders of any size are delivered over a modem. This policy cannot be set to a value less than 100 MB.

Maximum PTF order size over LAN. The maximum size in megabytes for a PTF order to be delivered electronically over the local area network (LAN). If -1 is specified, PTF orders of any size are delivered over the LAN. This policy is shipped with a default value of -1. This policy cannot be set to a value less than 100 MB (MB equals approximately 1 000 000 bytes).

Error Messages

The following messages may be sent from this function:

Message ID	Error Message Text
CPFE080	Maximum PTF order size not valid.
CPF0CC1	Error initializing the XML parser.
CPF24B4	Severe error while addressing parameter list.
CPF3CF1	Error code parameter not valid.
CPF3CF2	Error(s) occurred during running of &1 API.
CPF3C1E	Required parameter &1 omitted.
CPF3C21	Format name &1 is not valid.
CPF3C3A	Value for parameter &2 for API &1 not valid.
CPF9872	Program or service program &1 in library &2 ended. Reason code &3.

API introduced: V5R3 with PTF SI20372

Top | "Problem Management APIs," on page 1 | APIs by category

Start Problem Log Services (QsxStartProblemLogServices) API

Required Parameter Group:

1	Handle
Output	Binary(4)
2	Error code
I/O	Char(*)

Default Public Authority: *USE

Service Program: QSXSRVPL

Threadsafe: No

The Start Problem Log Services (QsxStartProblemLogServices) API sets up the environment to allow creating, changing, deleting, and retrieving problem log entries. The procedure performs the following functions:

- Opens the problem log files for update.
- Starts commitment control
- Returns a handle that must be supplied as a parameter by the using problem log APIs.

Only one instance of the problem log services may be started from a job. Attempting to start multiple instances of the problem log services will result in an error. Attempting to use one of the problem log APIs without the proper handle will also result in an error.

Authorities and Locks

API Public Authority
*USE

Required Parameter

Handle
OUTPUT; BINARY(4)

This provides a means of associating the problem log services that are started with subsequent problem log activities that will be performed.

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
CPF3C1E E	Required parameter &1 omitted.
CPF3C90 E	Literal value cannot be changed.
CPF3CF1 E	Error code parameter not valid.
CPF3CF2 E	Error(s) occurred during running of &1 API.
CPF7A86 E	Problem log services already started.
CPF9821 E	Not authorized to program &1 in library &2.
CPF9872 E	Program or service program &1 in library &2 ended. Reason code &3.
CPFA320 E	Pointer parameter is null.

API introduced: V3R1

Top | "Problem Management APIs," on page 1
APIs by category

Work with Problem (QPDWRKPB) API

Required Parameter Group:

1	Display panels	
Input	Char(10)	
2	Problem ID number	
Input	Char(10)	
3	Origin system	
Input	Char(20)	
4	Current problem status	
Input	Char(10)	
5	Requested problem statuses	
Input	Array of Char(10)	
6	Number of requested problem statuses	
Input	Binary(4)	
7	Service provider network identifier	
Input	Char(8)	
8	Service provider control point name	
Input	Char(8)	
9	Problem severity	
Input	Char(1)	
10	Error code	
I/O	Char(*)	

Optional Parameter Group:

11	Note text
Input	Char(*)
12	Length of note text
Input	Binary(4)

Default Public Authority: *USE

Threadsafe: No

The Work with Problem (QPDWRKPB) API uses a problem log entry to analyze and prepare a machine-detected hardware problem for reporting. Only local machine-detected problems that have an OPEN, READY, PREPARED, or SENT status can be analyzed and prepared for reporting. Remote problems that have an OPEN, READY, PREPARED, or SENT status can be prepared for reporting but cannot be analyzed. This API does not analyze or prepare user-detected problems.

If a machine-detected problem is analyzed, the problem analysis program associated with the problem is called to generate a problem analysis list identifying all the possible causes for the problem.

Required Parameter Group

Display panels
INPUT; CHAR(10)

Whether or not displays are shown during problem analysis. Valid values are:

**NO*	No displays are shown. **Note:** During problem analysis, if displays are usually shown for the type of hardware associated with the problem, then the Point of Failure list is saved. This list is only saved if no other list of causes currently associated with the problem exists.
**YES*	All displays are shown. This value is not allowed if the API is called in a batch job.

Problem ID number
INPUT; CHAR(10)

The number the system generates to identify a problem.

Origin system
INPUT; CHAR(20)

The node name of the origin system (the format is *network-ID.control-point-name*).

Current problem status
INPUT; CHAR(10)

The current status of the problem. If the problem is found to be in a status other than what is indicated, an error occurs. Valid values are:

**OPENED*	The problem was identified and a problem record was created.
**READY*	Problem analysis information has been added to the problem record.
**PREPARED*	The problem has been prepared for reporting.
**SENT*	The problem record was sent to a service provider, and the information needed to correct the problem was not returned.
**ANSWERED*	The problem has been answered.

Requested problem statuses
INPUT; ARRAY of CHAR(10)

The requested status for the problem.

Valid values are:

**READY*	Analyze the problem. Valid for local machine-detected problems only.
**PREPARED*	Prepare the problem for reporting. The service provider network identifier, service provider control point name, and problem severity parameters and the default contact database are used. **Note:** If you select *READY and *PREPARED, the final status is *PREPARED.

Number of requested problem statuses
INPUT; BINARY(4)

The number of statuses entered for requested problem statuses parameter.

Service provider network identifier
INPUT; CHAR(8)

The network identifier of the service provider system where the problem is to be sent.

Valid values are:

**NETATR*	The network identifier of this system. Use *NETATR if the control point name is *IBMSRV.

Service provider control point name
INPUT; CHAR(8)

The control point name of the service provider system where the problem is to be sent.

Valid values are:

**IBMSRV*	IBM service support. This value cannot be used if the problem has an *OPENED status unless you also have a requested problem status of *READY.
Name	The control point name.

Problem severity
INPUT; CHAR(1)

The severity of the problem.

Valid values are:

1	A high severity level in which there is a critical affect on operations. A severity 1 problem requires a service representative at the site, and the person solving the problem must work on the problem 24 hours a day until the problem is solved or circumvented. If the problem needs more information, patching, or the problem must be created again on the failing system, a service representative must be available to do these tasks immediately. It is not a severity 1 if these and any other tasks cannot be performed immediately.
2	A medium severity level in which you are able to use the system, but your operations are severely restricted by the problem.
3	A low severity level in which you are able to continue operations with some restrictions. The restrictions do not have a critical effect on your operations.
4	The severity level is minimal because the problem causes little or no effect to your operation, or you have found a way to circumvent the problem.

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

Note text
INPUT; CHAR(*)

The field used to include a note stating that a problem originated with the service director. The note will be included when the API is called.

Length of note text
INPUT; BINARY(4)

The length of the text for the note text field. The text field for the note text field can be up to 80 characters long.

Error Messages

Message ID	Error Message Text
CPF3C90 E	Literal value cannot be changed.
CPF3CF1 E	Error code parameter not valid.

Message ID	Error Message Text
CPF3CF2 E	Error(s) occurred during running of &1 API.
CPF7AA7 E	Problem &1 not found.
CPF7A9C E	Cannot work with the problem log at this time.
CPF7A9D E	Problem log object &1 is missing.
CPF7A93 E	Problem &2 currently in use by job &1.
CPF8C09 E	&1 not defined as a service provider.
CPF8C24 E	Error occurred while processing request.
CPF8C87 E	Service provider &1.&2 not found.
CPF9308 E	Unable to complete problem analysis. Reason code &1.
CPF931C E	Problem analysis results not recorded in problem log.
CPF9310 E	Problem analysis procedure was exited before it had completed.
CPF9313 E	Requested procedure is not allowed.
CPF932A E	Number of requested statuses is not valid.
CPF932B E	*IBMSRV not allowed as service provider for problems in OPENED status.
CPF9320 E	&1 to display panels is not valid.
CPF9321 E	Panels cannot be displayed in a batch job.
CPF9322 E	Current status &3 does not match problem status &4.
CPF9323 E	Current status &1 is not valid.
CPF9324 E	Problems on a remote system cannot be analyzed.
CPF9325 E	Problem &1 has a status that is not allowed.
CPF9326 E	Problem selected not allowed.
CPF9327 E	Problem analysis procedure was exited before it had completed.
CPF9328 E	Severity &1 is not valid.
CPF9329 E	Requested status &1 is not valid.
CPF7845 E	Error occurred while opening file &1.
CPF7846 E	Error while processing file &1 in library &2.
CPF7847 E	Error occurred while closing file &1 in library &2.
CPF7872 E	Program or service program &1 in library &2 ended. Reason code &3.

API introduced: V2R2

Top | "Problem Management APIs," on page 1
APIs by category

Concepts

These are the concepts for this category.

Key Groups for Problem Log APIs

Key Use for Problem Log APIs

This section describes keys applicable for the following Problem Log APIs:

- QsxAddProblemLogEntry
- QsxChangeProblemLogEntry
- QsxCreateProblemLogEntry
- QsxDeleteProblemLogEntry
- QsxRetrieveProblemLogEntry

Key utilization matrix

Key	API				
	Add	**Change**	**Create**	**Delete**	**Retrieve**
Group 0000 - General problem log entries					
1	Always	Always	Always	Always	Always
2	No	No	Yes	No	Yes
3	No	Yes	Yes	No	Yes
4	No	Yes	Yes	No	Yes
5	No	No	Yes	No	Yes
6	No	Yes	Yes	No	Yes
7	No	Yes	Yes	No	Yes
8	Yes	Yes	Yes	No	Yes
"Key Group 1000-Problem Description Entries" on page 63					
1000	No	Yes	Yes	No	Yes
1001	No	Yes	Yes	No	Yes
1002	No	Yes	Yes	No	Yes
1003	No	Yes	Yes	No	Yes
1004	No	Yes	Yes	No	Yes
1005	No	Yes	Yes	No	Yes
1006	No	Yes	Yes	No	Yes
1007	No	Yes	Yes	No	Yes
1008	No	Yes	Yes	No	Yes
1009	No	Yes	Yes	No	Yes
1010	No	Yes	Yes	No	Yes
1011	No	Yes	Yes	No	Yes
1012	No	Yes	Yes	No	Yes
1013	No	Yes	Yes	No	Yes
1014	No	Yes	Yes	No	Yes
1015	No	Yes	No	No	Yes
1016	No	Yes	No	No	Yes
"Key Group 2000-FRU Entries" on page 71					
2000	No	No	No	Yes	Yes
2001	Yes	No	Yes	No	No
2002	Yes	No	Yes	No	No
2003	Yes	No	Yes	No	No
2004	Yes	No	Yes	No	No
2005	Yes	No	Yes	No	No
2006	Yes	No	Yes	No	No
2007	Yes	No	Yes	No	No
2008	Yes	No	Yes	No	No
2009	Yes	No	Yes	No	No

Key	API				
	Add	Change	Create	Delete	Retrieve
“Key Group 3000-Text Entries” on page 77					
3000	No	No	No	No	Yes
3001	No	Yes	Yes	No	Yes
“Key Group 4000-Supporting data entries” on page 78					
4000	No	No	No	Yes	Yes
4001	Yes	No	Yes	No	No
4002	Yes	No	Yes	No	No
“Key Group 5000-Contact Entries” on page 80					
5000	No	No	No	No	Yes
5001	No	Yes	Yes	No	Yes
“Key Group 6000-Problem History Entries” on page 81					
6000	No	No	No	No	Yes
6001	Yes	No	Yes	No	No
“Key Group 7000-PTF Entries” on page 82					
7000	No	No	No	Yes	Yes
7001	Yes	Yes	Yes	Yes	Yes
7002	No	Yes	Yes	No	Yes
“Key Group 8000-Analyzed Error Entries” on page 84					
8000	No	No	No	No	Yes
“Key Group 9000-Logical Partition ID Entries” on page 84					
9000	No	No	No	No	Yes

Key Group 0000-General Problem Log Entries

This group is required for all problem entries.

This section contains the following keys:
- “Key 1-Problem log id” on page 60
- “Key 2-Problem type” on page 60
- “Key 3-Problem status” on page 60
- “Key 4-User assigned” on page 61
- “Key 5-Problem origin system” on page 61
- “Key 6-Operational data” on page 62
- “Key 7-Filter control” on page 63
- “Key 8-Answer codes” on page 63

For more details about the fields in the following table, see “Field Descriptions for Key Groups for Problem Log APIs” on page 85.

Key 1-Problem log id

Key 1 is required to identify the entry to which data will be added. Key 1 has the following uses:

- Defines whether the problem is being created for a local or remote problem.
- Provides the problem log identifier that is used with the Add, Change, Delete, or Retrieve Problem Log Entry APIs.

Note: The problem log output parameter provided on the Create Problem Log Entry API is returned in the key 1 format.

Offset			
Dec	**Hex**	**Type**	**Field**
0	0	BINARY(4)	Key
4	4	BINARY(4)	Key size
8	8	CHAR(31)	Problem log identifier
39	27	CHAR(1)	Reserved

Key 2-Problem type

This key is used to:

- Define the type of problem log entry
- Return the type of problem log entry retrieved

Offset			
Dec	**Hex**	**Type**	**Field**
0	0	BINARY(4)	Key
4	4	BINARY(4)	Key size
8	8	BINARY(4)	Problem type. See “Field Descriptions for Key Groups for Problem Log APIs” on page 85 for a description of the problem types.

Key 3-Problem status

Defines the status of the problem log. The problem statuses are OPENED, READY, SENT, ANSWERED, VERIFIED, PREPARED and CLOSED. PREPARED status implies that the problem log contains data that enables it to be sent to a service provider. The status is incremental. This means that the problem log entry contains the minimum level of data required for the problem to achieve such a status. PREPARED may be applied anytime after a problem has been opened and before it is closed.

Key 6001 is required with this key to record that the problem status has been changed. The status can be created, changed, or retrieved.

Offset			
Dec	**Hex**	**Type**	**Field**
0	0	BINARY(4)	Key
4	4	BINARY(4)	Key size
8	8	BINARY(4)	Problem status

Key 4-User assigned

Defines to whom the problem has been assigned.

This entry can be created, changed, or retrieved.

Offset			
Dec	Hex	Type	Field
0	0	BINARY(4)	Key
4	4	BINARY(4)	Key size
8	8	CHAR(10)	User assigned
18	12	CHAR(2)	Reserved

Key 5-Problem origin system

Defines the system on which this problem log entry originated. The system may be local (this system) or remote (another system). If the Create location field is set to local, the Create Problem Log Entry API automatically adds the following groups of fields:

- Origin system hardware description
- Origin system operating system

This entry can only be created and retrieved.

Offset			
Dec	Hex	Type	Field
0	0	BINARY(4)	Key
4	4	BINARY(4)	Key size
8	8	BINARY(4)	Create location
Note: The following fields Machine type length through Serial number pertain to the origin system hardware description.			
12	C	BINARY(4)	Machine type length
16	10	BINARY(4)	Model length
20	14	BINARY(4)	Feature length
24	18	BINARY(4)	Serial number length
28	1C	CHAR(16)	Machine type
44	2C	CHAR(16)	Model
60	3C	CHAR(16)	Feature
76	4C	CHAR(32)	Serial number
Note: The following fields Product ID length through Reserved pertain to the origin system operating system.			
108	6C	BINARY(4)	Product ID length
112	70	BINARY(4)	Version length
116	74	BINARY(4)	Release level length
120	78	BINARY(4)	Modification level length
124	7C	CHAR(15)	Product ID
139	8B	CHAR(5)	Version
144	90	CHAR(5)	Release level

Offset			
Dec	**Hex**	**Type**	**Field**
149	95	CHAR(5)	Modification level
154	9A	CHAR(2)	Reserved
156	9C	CHAR(13)	Create date and time
169	A9	CHAR(2)	Delta level

Key 6-Operational data

This key provides operational information about the problem entry.

All fields, except the Time added field and the When closed fields, can be created, changed, deleted, or retrieved. The time fields are added automatically by the Create and Change Problem Log Entry APIs, respectively.

Offset			
Dec	**Hex**	**Type**	**Field**
0	0	BINARY(4)	Key
4	4	BINARY(4)	Key size
8	8	BINARY(4)	Key control for key 6
12	C	BINARY(4)	Creator of entry
16	10	BINARY(4)	Alert status
20	14	BINARY(4)	Auto PAR
24	18	BINARY(4)	Auto notify
28	1C	CHAR(10)	APAR Library
Note: The following fields Code and Network address are received from the system.			
38	26	CHAR(1)	Code
39	27	CHAR(20)	Network address
Note: The following fields Code and Network address are sent to the system.			
59	3B	CHAR(1)	Code
60	3C	CHAR(20)	Network address
Note: The following fields Code and Network address are prepared for the system.			
80	50	CHAR(1)	Code
81	51	CHAR(20)	Network address
101	65	CHAR(13)	Date and time added
114	72	CHAR(13)	Date and time closed
127	7F	CHAR(1)	Reserved
128	80	BINARY(4)	Mode of analysis

Key 7-Filter control

Offset			
Dec	**Hex**	**Type**	**Field**
0	0	BINARY(4)	Key
4	4	BINARY(4)	Key size
8	8	BINARY(4)	Filter event
12	C	CHAR(10)	Filter name
22	16	CHAR(10)	Filter library name
32	20	CHAR(10)	Filter group assigned
42	2A	CHAR(2)	Reserved

Key 8-Answer codes

Contains the answer that was received when the problem was sent to a service provider.

Offset			
Dec	**Hex**	**Type**	**Field**
0	0	BINARY(4)	Key
4	4	BINARY(4)	Key size
8	8	BINARY(4)	Key control for key 8
12	C	BINARY(4)	Answer code assigned
16	10	BINARY(4)	Answer code returned
20	14	CHAR(5)	Problem number
25	19	CHAR(3)	Problem branch number
28	1C	CHAR(3)	Problem country number

Key Group 1000-Problem Description Entries

This group creates, changes, and retrieves problem description entries.

To locate the key of your need, click one of the following:
- “Key 1001-Problem Severity” on page 64
- “Key 1002-Problem Description Message” on page 64
- “Key 1003-Problem Creation Data” on page 64
- “Key 1004-Reporting Device” on page 65
- “Key 1005-Failing Resource” on page 65
- “Key 1006-Reporting Code” on page 66
- “Key 1007-Problem Analysis Data” on page 67
- “Key 1008-Fix Verification Status” on page 68
- “Key 1009-Fix Recovery Status” on page 68
- “Key 1010-Symptom String” on page 68
- “Key 1011-PTF Media Selection” on page 68
- “Key 1012-Problem Category” on page 69

- “Key 1013-Client Information” on page 69
- “Key 1014-First Failure Data Capture” on page 70
- “Key 1015-Query Status” on page 70
- “Key 1016-Hardware Location Information” on page 70

Key 1001-Problem Severity

This key defines the impact of the problem on the environment. This key is required for PREPARED status.

This entry can be created, changed, and retrieved.

For more details about the fields in the following table, see “Field Descriptions for Key Groups for Problem Log APIs” on page 85.

Offset			
Dec	**Hex**	**Type**	**Field**
0	0	BINARY(4)	Key
4	4	BINARY(4)	Key size
8	8	BINARY(4)	Problem severity

Key 1002-Problem Description Message

This key may be used where a message is used to describe the problem. If a message is not used, use key 3001 (text entry) to provide a description of the problem. Either key 1002 or 3001 is required. This key is required when the problem type is machine detected. This entry can be created, changed, or retrieved.

Offset			
Dec	**Hex**	**Type**	**Field**
0	0	BINARY(4)	Key
4	4	BINARY(4)	Key size
8	8	CHAR(7)	Message identifier
15	F	CHAR(10)	Message library name
25	19	CHAR(10)	Message file name
35	23	CHAR(1)	Reserved

Key 1003-Problem Creation Data

This is required for machine detected problem types and is optional for other problem types. This entry can be created, changed, or retrieved.

Offset			
Dec	**Hex**	**Type**	**Field**
0	0	BINARY(4)	Key
4	4	BINARY(4)	Key size
Note: The following fields Reference code through Reference code translate table library are part of the reference code description data.			
80	80	CHAR(2)	Reference code

Offset			
Dec	Hex	Type	Field
10	A	CHAR(10)	Reference code translate table identifier
20	14	CHAR(10)	Reference code translate table library
30	1E	CHAR(7)	Reference code description message
37	25	CHAR(10)	Reference code description file name
47	2F	CHAR(10)	Reference code description library name
57	39	CHAR(7)	Error code message identifier

Key 1004-Reporting Device

This key provides data that defines the machine that contains the failing hardware. This data is required for a problem to achieve READY status, since it contains the machine that a problem or PTF order will be reported against.

This entry can be created, changed, or retrieved.

Offset			
Dec	Hex	Type	Field
0	0	BINARY(4)	Key
4	4	BINARY(4)	Key size
8	8	BINARY(4)	Key control for key 1004
Note: The following fields Machine type length through Serial number pertain to the reporting device.			
12	C	BINARY(4)	Machine type length
16	10	BINARY(4)	Model length
20	14	BINARY(4)	Feature length
24	18	BINARY(4)	Serial number length
28	1C	CHAR(16)	Machine type
44	2C	CHAR(16)	Model
60	3C	CHAR(16)	Feature
76	4C	CHAR(32)	Serial number
108	6C	CHAR(12)	EC number

Key 1005-Failing Resource

This key contains data that defines the object that is failing. Hardware that can fail includes a machine, a feature, or a component of the machine. To an observer they might appear the same: they both have a type, a serial number, and a model. The major distinction is whether you have a maintenance contract.

For example, you can report a problem on a tape device 6366, but you cannot report a problem on an IOP feature number 2615. The 2615 is part of system machine type 9406. A problem can be reported against 9406 because it has a maintenance contract. This entry can be created, changed, or retrieved. Where a program object is failing, the product data is also added. Otherwise it must be blank.

Offset			
Dec	**Hex**	**Type**	**Field**
0	0	BINARY(4)	Key
4	4	BINARY(4)	Key size
8	8	BINARY(4)	Key control for key 1005
12	C	BINARY(4)	Type of hardware
Note: The following fields Machine type length to Serial number pertain to the failing device structure.			
16	10	BINARY(4)	Machine type length
20	14	BINARY(4)	Model length
24	18	BINARY(4)	Feature length
28	1C	BINARY(4)	Serial number length
32	20	CHAR(16)	Machine type
48	30	CHAR(16)	Model
64	40	CHAR(16)	Feature
80	50	CHAR(32)	Serial number
Note: The following fields Product ID length through Reserved pertain to the failing product structure.			
112	70	BINARY(4)	Product ID length
116	74	BINARY(4)	Version length
120	78	BINARY(4)	Release level length
124	7C	BINARY(4)	Modification level length
128	80	CHAR(15)	Product ID
143	8F	CHAR(5)	Version
148	94	CHAR(5)	Release level
153	99	CHAR(5)	Modification level
158	9E	CHAR(2)	Reserved
160	A0	CHAR(4)	Instruction
164	A4	CHAR(20)	Hierarchy
184	B8	CHAR(10)	Resource name
194	C2	CHAR(4)	Error log identifier
198	C6	CHAR(10)	Program

Key 1006-Reporting Code

Data that defines the program object that is failing or the object against which the problem will be reported. For example, the licensed internal code of a feature, such as an IOA, is the product on which the problem will be reported. It is the program object with a maintenance contract.

Offset			
Dec	Hex	Type	Field
0	0	BINARY(4)	Key
4	4	BINARY(4)	Key size
8	8	BINARY(4)	Key control for key 1006
Note: The following fields Product ID length through Reserved pertain to the reporting product description.			
12	C	BINARY(4)	Product ID length
16	10	BINARY(4)	Version length
20	14	BINARY(4)	Release level length
24	18	BINARY(4)	Modification level length
28	1C	CHAR(15)	Product ID
43	2B	CHAR(5)	Version
48	30	CHAR(5)	Release level
53	35	CHAR(5)	Modification level
58	3A	CHAR(2)	Reserved
60	3C	CHAR(10)	Program
70	46	CHAR(4)	Probe
74	4A	CHAR(2)	Reserved

Key 1007-Problem Analysis Data

This key contains the post problem analysis results. The reference code description data defines the program that isolated the error and provides a reference to an object that contains detailed data describing the failure.

This key is required to move a machine detected problem to READY status. It is optional with other problem types. The entry can be created, changed, or retrieved.

Offset			
Dec	Hex	Type	Field
0	0	BINARY(4)	Key
4	4	BINARY(4)	Key size
8	8	BINARY(4)	Number of times analyzed
12	C	BINARY(4)	Isolation status
16	10	CHAR(8)	PDP
Note: The following fields Reference code through Reference code translate table library pertain to the reference code description data.			
24	18	CHAR(2)	Reference code
26	1A	CHAR(10)	Reference code translate table identifier
36	24	CHAR(10)	Reference code translate table library
46	2E	CHAR(2)	Exit point of the PDP

Key 1008-Fix Verification Status

The key that data that defines the status of the verification attempt.

The problem must be in SENT or ANSWERED status to append this data. This entry can be created, changed, or retrieved.

Offset			
Dec	**Hex**	**Type**	**Field**
0	0	BINARY(4)	Key
4	4	BINARY(4)	Key size
8	8	BINARY(4)	Verification status
12	C	CHAR(8)	PDP

Key 1009-Fix Recovery Status

This key contains data that defines status of the recovery attempt.

The problem must be in SENT or ANSWERED status to append this data. This entry can be created, changed, or retrieved.

Offset			
Dec	**Hex**	**Type**	**Field**
0	0	BINARY(4)	Key
4	4	BINARY(4)	Key size
8	8	BINARY(4)	Recovery status
12	C	CHAR(8)	PDP

Key 1010-Symptom String

This key contains data that is used to search a data base for the existence of a problem.

The problem must be READY status to append this data. A problem cannot be moved to PREPARED status without this key. This entry can be created, changed, or retrieved. It is not allowed on problem type 3, PTF order.

Offset			
Dec	**Hex**	**Type**	**Field**
0	0	BINARY(4)	Key
4	4	BINARY(4)	Key size
8	8	CHAR(256)	Symptom
Note: The first character position of this symptom field may not contain a blank.			

Key 1011-PTF Media Selection

This key contains data that is used to define the type of media on which a PTF should be delivered. The type of media is defined by the media type and the machine type on which the media is installed.

Note: If the machine type and model are unknown, zeros must be used for these fields.

A problem cannot be moved to PREPARED status without this key.

Offset			
Dec	**Hex**	**Type**	**Field**
0	0	BINARY(4)	Key
4	4	BINARY(4)	Key size
8	8	BINARY(4)	Machine type length
12	C	BINARY(4)	Model length
16	10	BINARY(4)	Media type
20	14	CHAR(16)	Machine type
36	24	CHAR(16)	Model

Key 1012-Problem Category

This key contains data that is used to define the category of a problem.

Offset			
Dec	**Hex**	**Type**	**Field**
0	0	BINARY(4)	Key
4	4	BINARY(4)	Key size
8	8	BINARY(4)	Problem category

Key 1013-Client Information

This key contains data that defines the failing software on a personal computer.

Offset			
Dec	**Hex**	**Type**	**Field**
0	0	BINARY(4)	Key
4	4	BINARY(4)	Key size
8	8	BINARY(4)	Path Name length
12	C	BINARY(4)	Product ID length
16	10	BINARY(4)	Version length
20	14	BINARY(4)	Program length
24	18	BINARY(4)	Function length
28	1C	BINARY(4)	Client ID length
32	20	BINARY(4)	Contact information length
36	24	CHAR(1024)	Path Name
292	124	CHAR(64)	Product ID
356	164	CHAR(64)	Version
420	1A4	CHAR(64)	Program
484	1E4	CHAR(64)	Function

Offset			
Dec	Hex	Type	Field
548	224	CHAR(256)	Client ID
804	324	CHAR(256)	Contact information
1060	424	CHAR(20)	Address

Key 1014-First Failure Data Capture

This key contains data that is used to indicate the number of times a problem has recurred. The data contains the program that detected the failure and a description of the product.

Offset			
Dec	Hex	Type	Field
0	0	BINARY(4)	Key
4	4	BINARY(4)	Key size
8	8	BINARY(4)	Key control for key 1014
12	C	BINARY(4)	Count
16	10	BINARY(4)	Object name length
20	14	CHAR(256)	Object name
Note: The following fields Product ID length through Reserved pertain to product data.			
276	114	BINARY(4)	Product ID length
280	118	BINARY(4)	Version length
284	11C	BINARY(4)	Release level length
288	120	BINARY(4)	Modification level length
292	124	CHAR(15)	Product ID
307	133	CHAR(5)	Version
312	138	CHAR(5)	Release level
317	13D	CHAR(5)	Modification level
322	142	CHAR(2)	Reserved

Key 1015-Query Status

An indicator of the results of a query of the problem log status.

Offset			
Dec	Hex	Type	Field
0	0	BINARY(4)	Key
4	4	BINARY(4)	Key size
8	8	BINARY(4)	Query status

Key 1016-Hardware Location Information

This key indicates the physical location of the hardware for frame ID and device locations.

Offset			
Dec	**Hex**	**Type**	**Field**
0	0	BINARY(4)	Key
4	4	BINARY(4)	Key size
8	8	CHAR(4)	Frame ID location
12	C	CHAR(5)	Device location
16	10	CHAR(4)	Card location

Key Group 2000-FRU Entries

This key group provides information about field replaceable unit (FRU). This group can only be used with machine-detected problem types. Keys 2001 through 2009 use a header to define the FRU type, probability, FRU code, and message identifier for the FRU.

Click one of the following to find your key:

- "Key 2000-Number of FRU Entries to Work with"
- "Key 2001-Device FRU Type"
- "Key 2002-Code FRU Type" on page 73
- "Key 2003-Media FRU Type" on page 73
- "Key 2004-User FRU Type" on page 74
- "Key 2005-FRU Name" on page 75
- "Key 2006-Attached FRU" on page 75
- "Key 2007-Configuration FRU" on page 75
- "Key 2008-General FRU" on page 76
- "Key 2009-Channel Attached FRU" on page 76

Key 2000-Number of FRU Entries to Work with

This key deletes or retrieves all FRU entries or all FRU entries of a class.

For more details about the fields in the following table, see "Field Descriptions for Key Groups for Problem Log APIs" on page 85.

Offset			
Dec	**Hex**	**Type**	**Field**
0	0	BINARY(4)	Key
4	4	BINARY(4)	Key size
8	8	BINARY(4)	Class of FRU
12	C	BINARY(4)	FRU count

Key 2001-Device FRU Type

This defines the data required to create a FRU entry for a device or feature. Device here can also be a feature code. The device data defines the device or feature.

Offset			
Dec	Hex	Type	Field
0	0	BINARY(4)	Key
4	4	BINARY(4)	Key size
Note: The following fields Class of FRU through Reserved pertain to the FRU header.			
8	8	BINARY(4)	Class of FRU
12	C	BINARY(4)	Probability of fix
16	10	CHAR(4)	FRU code
20	14	CHAR(7)	FRU description message ID
27	1B	CHAR(1)	Reserved
28	1C	BINARY(4)	Device location text length
32	20	BINARY(4)	Coded character set identifier
36	24	BINARY(4)	Rack serial number length
Note: The following fields Machine type length through Serial number pertain to device data.			
40	28	BINARY(4)	Machine type length
44	2C	BINARY(4)	Model length
48	30	BINARY(4)	Feature length
52	34	BINARY(4)	Serial number length
56	38	CHAR(16)	Machine type
72	48	CHAR(16)	Model
88	58	CHAR(16)	Feature
104	68	CHAR(32)	Serial number
136	88	CHAR(7)	Document reference message ID
143	8F	CHAR(256)	Device location text
399	18F	CHAR(10)	Resource name
409	199	CHAR(10)	Device name
419	1A3	CHAR(32)	Rack serial number
451	1C3	CHAR(2)	Card position
453	1C5	CHAR(2)	DSA bus number
455	1C7	CHAR(4)	Unit address
459	1CB	CHAR(2)	Port
461	1CD	CHAR(3)	Reserved
464	1D0	BINARY(4)	Device type
Note: The following fields, Transport type through Dependent address 5, pertain to RISC device data.			
468	1D4	BINARY(4)	Transport type
472	1D8	BINARY(4)	Bus number
476	1DC	BINARY(4)	Card number
480	1E0	BINARY(4)	Board number
484	1E4	BINARY(4)	Address type
488	1E8	BINARY(4)	I/O bus address
492	1EC	BINARY(4)	Dependent address 2
496	1F0	BINARY(4)	Dependent address 3

Offset			
Dec	**Hex**	**Type**	**Field**
500	1F4	BINARY(4)	Dependent address 4
504	1F8	BINARY(4)	Dependent address 5

Key 2002-Code FRU Type

This defines the data required to create a FRU entry for code. Code may be a product, a program, or a module.

Offset			
Dec	**Hex**	**Type**	**Field**
0	0	BINARY(4)	Key
4	4	BINARY(4)	Key size
Note: The following fields Class of FRU through Reserved pertain to the FRU header.			
8	8	BINARY(4)	Class of FRU
12	C	BINARY(4)	Probability of fix
16	10	CHAR(4)	FRU code
20	14	CHAR(7)	FRU description message ID
27	1B	CHAR(1)	Reserved
Note: The following fields Product ID length through Reserved pertain to product data.			
28	1C	BINARY(4)	Product ID length
32	20	BINARY(4)	Version length
36	24	BINARY(4)	Release level length
40	28	BINARY(4)	Modification level length
44	2C	CHAR(15)	Product ID
59	3B	CHAR(5)	Version
64	40	CHAR(5)	Release level
69	45	CHAR(5)	Modification level
74	4A	CHAR(2)	Reserved
76	4C	CHAR(4)	Primary function group
80	50	CHAR(4)	Secondary function group
84	54	CHAR(10)	Module name
94	5E	CHAR(7)	Document reference message ID
101	65	CHAR(3)	Reserved

Key 2003-Media FRU Type

This defines the data required to create a FRU entry for media. The device data defines the device on which the media, such as tape or diskette, was installed.

Offset		Type	Field
Dec	Hex		
0	0	BINARY(4)	Key
4	4	BINARY(4)	Key size
Note: The following fields Class of FRU through Reserved pertain to the FRU header.			
8	8	BINARY(4)	Class of FRU
12	C	BINARY(4)	Probability of fix
16	10	CHAR(4)	FRU code
20	14	CHAR(7)	FRU description message ID
27	1B	CHAR(1)	Reserved
Note: The following fields Machine length type through Serial number pertain to the device data.			
28	1C	BINARY(4)	Machine type length
32	20	BINARY(4)	Model length
36	24	BINARY(4)	Feature length
40	28	BINARY(4)	Serial number length
44	2C	CHAR(16)	Machine type
60	3C	CHAR(16)	Model
76	4C	CHAR(16)	Feature
92	5C	CHAR(32)	Serial number
124	7C	CHAR(7)	Document reference message ID
131	83	CHAR(10)	Resource name
141	8D	CHAR(8)	Volume ID
149	95	CHAR(3)	Reserved

Key 2004-User FRU Type

This defines the data required to define a problem resulting from an user action.

Offset		Type	Field
Dec	Hex		
0	0	BINARY(4)	Key
4	4	BINARY(4)	Key size
Note: The following fields Class of FRU through Reserved pertain to the FRU header.			
8	8	BINARY(4)	Class of FRU
12	C	BINARY(4)	Probability of fix
16	10	CHAR(4)	FRU code
20	14	CHAR(7)	FRU description message ID
27	1B	CHAR(1)	Reserved
28	1C	CHAR(7)	Document reference message ID
35	23	CHAR(1)	Reserved

Key 2005-FRU Name

This defines the data required to create a list of up to six parts that could be failing. The parts are identified by their part numbers and location.

Offset			
Dec	**Hex**	**Type**	**Field**
0	0	BINARY(4)	Key
4	4	BINARY(4)	Key size
Note: The following fields Class of FRU through Reserved pertain to the FRU header.			
8	8	BINARY(4)	Class of FRU
12	C	BINARY(4)	Probability of fix
16	10	CHAR(4)	FRU code
20	14	CHAR(7)	FRU description message ID
27	1B	CHAR(1)	Reserved
28	1C	CHAR(7)	Document reference message ID
35	23	CHAR(25)	Part location
60	3C	CHAR(6)(12)	Part number array

Key 2006-Attached FRU

This defines the data required to create a list of up to six parts that could be failing. The parts are identified by their part numbers and location.

This FRU defines parts that are attached to I/O adapters or I/O processors.

Offset			
Dec	**Hex**	**Type**	**Field**
0	0	BINARY(4)	Key
4	4	BINARY(4)	Key size
Note: The following fields Class of FRU through Reserved pertain to the FRU header.			
8	8	BINARY(4)	Class of FRU
12	C	BINARY(4)	Probability of fix
16	10	CHAR(4)	FRU code
20	14	CHAR(7)	FRU description message ID
27	1B	CHAR(1)	Reserved
28	1C	CHAR(7)	Document reference message ID
35	23	CHAR(25)	Part location
60	3C	CHAR(6)(12)	Part number array

Key 2007-Configuration FRU

This key defines an error in the configuration of a device. It provides the name of a panel that may be displayed defining a problem.

Offset		Type	Field
Dec	Hex		
0	0	BINARY(4)	Key
4	4	BINARY(4)	Key size
Note: The following fields Class of FRU through Reserved pertain to the FRU header.			
8	8	BINARY(4)	Class of FRU
12	C	BINARY(4)	Probability of fix
16	10	CHAR(4)	FRU code
20	14	CHAR(7)	FRU description message ID
27	1B	CHAR(1)	Reserved
28	1C	BINARY(4)	Coded character set identifier
32	20	BINARY(4)	Replacement text length
36	24	CHAR(8)	Screen identifier
44	2C	CHAR(30)	Replacement text
74	4A	CHAR(2)	Reserved

Key 2008-General FRU

This defines a FRU that is not of one of the other classes of FRUs. It provides the name of a panel that may be displayed defining a problem.

Offset		Type	Field
Dec	Hex		
0	0	BINARY(4)	Key
4	4	BINARY(4)	Key size
Note: The following fields Class of FRU through Reserved pertain to the FRU header.			
28	1C	BINARY(4)	Class of FRU
32	20	BINARY(4)	Probability of fix
36	24	CHAR(4)	FRU code
40	28	CHAR(7)	FRU description message ID
47	2F	CHAR(1)	Reserved
48	30	BINARY(4)	Coded character set identifier
52	34	BINARY(4)	Replacement text length
56	38	CHAR(8)	Screen identifier
64	40	CHAR(30)	Replacement text
94	5E	CHAR(2)	Reserved

Key 2009-Channel Attached FRU

Offset		Type	Field
Dec	Hex		
0	0	BINARY(4)	Key

Offset			
Dec	**Hex**	**Type**	**Field**
4	4	BINARY(4)	Key size
Note: The following fields Class of FRU through Reserved pertain to the FRU header.			
8	8	BINARY(4)	Class of FRU
12	C	BINARY(4)	Probability of fix
16	10	CHAR(4)	FRU code
20	14	CHAR(7)	FRU description message ID
27	1B	CHAR(1)	Reserved
28	1C	CHAR(7)	Document reference message ID
35	23	CHAR(4)	Fault symptom code
39	27	CHAR(32)	Sense bytes
71	47	CHAR(1)	Reserved

Key Group 3000-Text Entries

This key group creates, retrieves, and changes problem text entries. It provides access to text that defines, describes, or tracks a problem.

To get to the key of your need, click one of the following:
- "Key 3000-Text Entry"
- "Key 3001-Text Entry" on page 78

Key 3000-Text Entry

Retrieves text about a problem. Either all text associated with the problem or specified text can be retrieved. The text types associated with the problem are:
- 80 character title, limit to one entry

  This entry provides users with a means of describing a problem in their own words. This appears on the problem list panel.
- Long problem description.

  A detailed description of the problem.
- Problem status

  Used to provide a means of tracking a problem until it is resolved, especially tracking what the support organization is doing to resolve the problem.
- Private notes

  Provides an area to keep notes about a problem that will not be made public. These notes are not sent to another system.
- Associated problem data

  This area is for general use and can be tailored to the needs of the users.

For more details about the fields in the following table, see "Field Descriptions for Key Groups for Problem Log APIs" on page 85.

Offset		Type	Field
Dec	Hex		
0	0	BINARY(4)	Key
4	4	BINARY(4)	Key size
8	8	BINARY(4)	Text type
12	C	BINARY(4)	Text count

Key 3001-Text Entry

Allows the user to create or change data about a problem. The user is responsible for the content and format.

To create a text entry, provide the length of text to add. The text is referenced by a pointer and the coded character set identifier. A pointer, defined in key 3001 (Text entry), points to the beginning of the data.

To change the data, a retrieve, although not required, should be performed first. Data provided on the change API overlays the data previously in the entry. The data is changed by providing the data as done in a create. To effectively delete the data set, set Text length to 0. This entry can be created, changed, or retrieved.

Offset		Type	Field
Dec	Hex		
0	0	BINARY(4)	Key
4	4	BINARY(4)	Key size
8	8	BINARY(4)	Text type
12	C	BINARY(4)	Coded character set identifier
16	10	BINARY(4)	Text length
20	14	CHAR(12)	Reserved
32	20	POINTER(SPP)	Pointer to the text

Key Group 4000-Supporting data entries

This key group maintains a list of files that contain supplemental data about a problem. The data is contained in spooled or database files. The name and location of the files is maintained by this key group.

To get to the key of your need, click one of the following:

- “Key 4000-Supporting Data Entries”
- “Key 4001-Spooled File Data” on page 79
- “Key 4002-File Data” on page 79

Key 4000-Supporting Data Entries

This key retrieves and deletes all entries or all entries of a type, spooled or database files, associated with a specific problem. Spooled files are processed using key 4001 and database files are processed using key 4002. Deleting a specific entry is not supported. This entry can be used by the delete and retrieve API.

Offset		Type	Field
Dec	**Hex**		
0	0	BINARY(4)	Key
4	4	BINARY(4)	Key size
8	8	BINARY(4)	File type
12	C	BINARY(4)	File count

Key 4001-Spooled File Data

This key contains the name of a spooled file that is associated with the problem log entry.

This key is used to add or create an entry. It is also used to return the results of a retrieve operation.

To add or create an entry, use this key to define each spooled file to be associated with the problem. New entries are added to the file.

To change an entry it must be deleted first then a new one added.

A retrieve is done by passing key 4000 and defining type 1. All spooled file entries are returned, a key 4001 (spooled file data) for each. The entry is used by the Add and Create Problem Log Entry APIs.

Offset		Type	Field
Dec	**Hex**		
0	0	BINARY(4)	Key
4	4	BINARY(4)	Key size
8	8	BINARY(4)	File number
12	C	CHAR(10)	Job name
22	16	CHAR(10)	User
32	20	CHAR(6)	Job Number
38	26	CHAR(10)	File Name

Key 4002-File Data

This key contains the name of a data base file that is associated with the problem log entry.

This key is used to add or create an entry. It is also used to return the results of a retrieve operation.

To add or create an entry, use key 4002 (file data) to define each spooled file to be associated with the problem. New entries are added to the file.

To change an entry it must be deleted first then a new ones added.

A retrieve is done by passing key 4000 and defining type 2. All data base file entries are returned, a key 4002 (file data) for each.

Offset		Type	Field
Dec	**Hex**		
0	0	BINARY(4)	Key

Offset			
Dec	Hex	Type	Field
4	4	BINARY(4)	Key size
8	8	CHAR(10)	File name
18	12	CHAR(10)	File library name
28	1C	CHAR(10)	File member name
38	26	CHAR(2)	Reserved

Key Group 5000-Contact Entries

This key group provides information about the contact.

This section contains the following keys:
- "Key 5000-Contact entries"
- "Key 5001-Contact Information"

Key 5000-Contact entries

Allows the retrieval of contact information, local, remote, or both. A key 5001 (Contact information) entry is returned for each of the contact entries. This can be used by the Retrieve Problem Log Entry API.

For more details about the fields in the following table, see "Field Descriptions for Key Groups for Problem Log APIs" on page 85.

Offset			
Dec	Hex	Type	Field
0	0	BINARY(4)	Key
4	4	BINARY(4)	Key size
8	8	BINARY(4)	Contact type
12	C	BINARY(4)	Contact information count

Key 5001-Contact Information

Allows creating or changing a contact entry. To create or change an entry:
- Provide the type of entry to create or change
- Set the key control to define the field to process.

  The control values are:

1	NLV
2	Corporation name
4	Contact name
8	Primary contact phone number
16	Help desk or pager number
32	Address
64	CCSID
128	Primary FAX contact phone number
256	Alternative FAX contact phone number

To process multiple fields sum the value of the fields to be processed.

- Provide the data to be added to the field. Enter a blank to delete the contents of a field.

Offset			
Dec	**Hex**	**Type**	**Field**
0	0	BINARY(4)	Key
4	4	BINARY(4)	Key size
8	8	BINARY(4)	Key control for key 5001
12	C	BINARY(4)	Contact type
16	10	BINARY(4)	Coded character set identifier
20	14	CHAR(4)	National language version
24	18	CHAR(36)	Corporation name
60	3C	CHAR(36)	Name of contact
96	60	CHAR(30)	Primary phone number
126	7E	CHAR(30)	Help desk or pager number
156	9C	CHAR(30)	Primary FAX number
186	BA	CHAR(30)	Alternative FAX number
Note: The following fields Address line 1 through Postal code pertain to the postal address.			
216	D8	CHAR(36)	Address line 1
252	FC	CHAR(36)	Address line 2
288	120	CHAR(36)	Address line 3
324	144	CHAR(36)	City or state
360	168	CHAR(20)	Country or region
380	17C	CHAR(12)	Postal code

Key Group 6000-Problem History Entries

This key group provides problem history structures.

This section includes the following keys:

- "Key 6000-History Information"
- "Key 6001-History Information" on page 82

Key 6000-History Information

This key retrieves all or the last history entry. Key 6001 (history information) is returned for each history entry. Entries are returned starting with the latest entry.

For more details about the fields in the following table, see "Field Descriptions for Key Groups for Problem Log APIs" on page 85.

Offset			
Dec	**Hex**	**Type**	**Field**
0	0	BINARY(4)	Key
4	4	BINARY(4)	Key size
8	8	BINARY(4)	History type

Offset			
Dec	**Hex**	**Type**	**Field**
12	C	BINARY(4)	History count

Key 6001-History Information

History entries should be added to the file in logical event sequence and must be added each time the problem log entry is created, changed, or elements are deleted. The create and add APIs add the entries in the sequence the key 6001 (history information) are supplied to the API. No verification is made of the logical order of the events. All entries that are added in the context of one API call have the same date and time. The API adds the date and time.

Once entered the event may not be changed or deleted. Change control is provided to allow optional data, change request name, and change request number to be added when needed.

Offset			
Dec	**Hex**	**Type**	**Field**
0	0	BINARY(4)	Key
4	4	BINARY(4)	Key size
8	8	BINARY(4)	Key control for key 6001
12	C	BINARY(4)	History type
16	10	CHAR(13)	Event date and time
29	1D	CHAR(10)	User ID
39	27	CHAR(10)	Change request name
49	31	CHAR(6)	Change request number
55	37	CHAR(1)	Reserved

Key Group 7000-PTF Entries

This key group provides program temporary fix (PTF) information.

This section contains the following keys:
- "Key 7000-PTF Entry"
- "Key 7001-PTF ID" on page 83
- "Key 7002-PTF ID" on page 84

Key 7000-PTF Entry

Allows an user to retrieve or delete all PTF entries.

On a retrieve operation it defines the number of entries returned on a retrieve operation.

On a delete operation, all the PTF entries are deleted. Number of entries has no significance during delete.

For more details about the fields in the following table, see "Field Descriptions for Key Groups for Problem Log APIs" on page 85.

Offset		Type	Field
Dec	Hex		
0	0	BINARY(4)	Key
4	4	BINARY(4)	Key size
8	8	BINARY(4)	PTF count

Key 7001-PTF ID

This key defines the program temporary fix (PTF) identifier. On add or create operations, all fields must be filled in.

On a retrieve operation, this key defines which PTF to retrieve. A PTF is identified by the PTF ID, product, version, release, and modification.

PTF entries are always added to the end of the list.

To change a PTF entry, the key control should be used to identify the field being changed. The PTF ID may not be changed.

Note: Ensure that the correct PTF entry is being changed. The SNDPTFORD command creates entries that use special values for the product data. Non-IBM products may use the same PTF ID for different releases or different vendors may use the same PTF ID. It may be necessary to retrieve, delete, and add new PTF entries where there are multiple PTFs with the same PTF ID, but different product data, are encountered. This exposure only exists with non-IBM PTFs since IBM PTFs have unique PTF identifiers.

Offset		Type	Field
Dec	Hex		
0	0	BINARY(4)	Key
4	4	BINARY(4)	Key size
8	8	BINARY(4)	Key control for key 7001
12	C	BINARY(4)	PTF status
16	10	BINARY(4)	Sent
20	14	BINARY(4)	PTF ID length
24	18	CHAR(20)	PTF ID
Note: The following fields Product ID length through Reserved pertain to the product data.			
44	2C	BINARY(4)	Product ID length
48	30	BINARY(4)	Version length
52	34	BINARY(4)	Release level length
56	38	BINARY(4)	Modification level length
60	3C	CHAR(15)	Product ID
75	4B	CHAR(5)	Version
80	50	CHAR(5)	Release level
85	55	CHAR(5)	Modification level
90	5A	CHAR(2)	Reserved
92	5C	CHAR(2)	Minimum level
94	5E	CHAR(2)	Maximum level

Offset			
Dec	**Hex**	**Type**	**Field**
96	60	CHAR(1)	PTF image

Key 7002-PTF ID

On a create operation, all fields must be provided.

On a change operation, only the fields identified by the Key control field are processed.

On a retrieve operation, the PTF ordering options are returned.

Offset			
Dec	**Hex**	**Type**	**Field**
0	0	BINARY(4)	Key
4	4	BINARY(4)	Key size
8	8	BINARY(4)	Key control for key 7002
12	C	BINARY(4)	PTF order type
16	10	BINARY(4)	Option
20	14	BINARY(4)	Reorder
24	18	BINARY(4)	Delivery
28	1C	BINARY(4)	Check
32	20	BINARY(4)	Delivery Format
36	24	CHAR(64)	Image directory
100	64	CHAR(10)	Image prefix

Key Group 8000-Analyzed Error Entries

This key group provides analyzed error flag information.

Key 8000-Analyzed Error Flag: This key retrieves a value that indicates whether SLIC analyzed the problem.

Offset			
Dec	**Hex**	**Type**	**Field**
0	0	BINARY(4)	Key
4	4	BINARY(4)	Key size
8	8	BINARY(4)	Analyzed error flag

Key Group 9000-Logical Partition ID Entries

This key group provides logical partition ID information.

Key 9000-Logical Partition ID: This key retrieves the current logical partition ID on the physical machine.

Offset			
Dec	Hex	Type	Field
0	0	BINARY(4)	Key
4	4	BINARY(4)	Key size
8	8	CHAR(1)	Logical partition ID

Field Descriptions for Key Groups for Problem Log APIs

Address. Internet address of the client in dotted form. For example: 95.5.123.11.

Address line 1. The first line of the address.

Address line 2. The second line of the address.

Address line 3. The third line of the address.

Address type. The format of the unit address, which is numeric value that represents the hardware type. Valid values are as follows:

1	Communications resource
2	Storage resource
3	Workstation resource
4	Auxiliary processor resource
5	Library resource

Alert status. Valid values are:

0	Problem not alertable
1	No alert pending
2	Alert pending

Alternative FAX number. The backup FAX number to receive problem-related data.

Analyzed error flag. Indicates whether the error has been analyzed by SLIC.

Answer code assigned. The code that is assigned corresponds to a message that describes the answer given to the problem. The values are:

-1	No code assigned
0	Fixes sent
1	Fixes mailed
2	Fixes to be sent later
3	Fix cover letter only
4	Fixes not available
5	Fixes already on system
6	Not ordered
7	Fixes ordered or on system
8	All fixes on order
9	Exceeds mailing limit
10	Exceeds transmit limit
11	Exceeds limit for cover letter order

12	Support center notified
13	Documentation error
14	Failing product not entitled
15	Service requester not entitled
16	Reporting system not entitled
17	Entries out of order

Answer code returned. The code that is assigned corresponds to a message that describes the answer given the service requester regarding the problem. See the answer code assigned field for list of values.

APAR library. The name of the library containing the saved APAR data for this problem. The library, if present, contains spooled and database files. This data is collected automatically or by the Restore APAR Data (RSTAPARDTA) or Save APAR Data (SAVAPARDTA) commands. The library is deleted when the problem log entry is deleted.

Auto PAR. Defines if problem analysis procedures were automatically run for this problem.

0	Problem analysis not done automatically.
1	Problem analysis done automatically.

Auto notify. Defines if the problem has been automatically reported to a service provider.

0	Notify not done automatically.
1	Notify done automatically.

Board number. The number of the back plane card on this bus.

Bus number. The number of the bus.

Card location. The physical location of the card.

Card number. The number derived from the slot number (the logical address is assigned to the card slot).

Card position. Physical location where the device or feature is plugged into the bus.

Change request name. The name assigned, by the user, when submitting a change request.

Change request number. The sequence number of the change request.

Check. Indicates whether checking is performed on the service requester system to determine if PTFs are ordered based on whether or not the PTF product is installed or supported. Possible values are:

**NO*	The PTFs specified on the PTF order list are ordered even when the PTF product is not installed or supported on the service requester.
**YES*	The PTFs specified on the PTF order list parameter are ordered only if the PTF product is installed or supported on the service requester.

Note: **NO* must be specified when *1 (Cover letter only)* is specified for PTF order type.

City or state. The city or state of the postal address.

Class of FRU. The class of FRU entries to process. The values are:

0	All FRU classes
1	Point of failure
2	Partial isolation
3	Isolation
4	Verification
5	Recovery
6	Answer

All FRUs can only be used on a retrieve operation.

Client ID. Name of the client.

Client ID length. Length of the client ID data.

Code. A code that defines the network address type.

A	APPN
I	Internet
R	*IBMSRV

Coded character set identifier (CCSID). A code that describes the character set of the text. This value should be changed each time data is written and the value must agree with the CCSID of the data. If this value is 0 on a create operation, the API uses the job CCSID.

Contact information. Data describing the PC contact.

Contact information count. The number of 5001 keys that are returned by the retrieve operation.

Contact information length. Length of the contact information data. If it is a local contact information record, it is the local corporation name, or else it is the remote system corporation name.

Contact type. Origin of contact information, local or remote. The values are:

0	Contact information of all systems.
1	Contact information of the local system.
2	Contact information of the system on which the problem was created.

Corporation name. Name of company that depends on the entry type.

Count. Number of times the problem has been detected.

Country or region. The country or region of the postal address.

Create date and time. Time the problem log entry was created and added by the API. It is in format CYYMMDDHHMMSS. Ignored if the create was local.

Create location. Defines where problem was created. The values are:

1	Local
2	Remote

Creator of entry. Defines the function that created the entry.

-1	Not defined
0	Alert
1	FFDC, first failure data capture
2	FAST
3	General
4	PWSI

Date and time added. Date and time the problem log entry was added. This is the time that the problem was added to this systems problem log. This field is only valid for the QsxRetrieveProblemLogEntry API.

This is entered by the create API when the problem is added to this system. The time added field cannot be changed once entered, but it can be retrieved.

Date and time closed. Date and time the problem log entry was closed.

This field is changed when the user selects close on the Work with Problem display or uses the Change Problem Log API. This field can be retrieved, but it cannot be changed.

Delivery. Defines whether the PTF will be delivered by mail or electronically.

0	Deliver by mail or electronically.
1	Deliver electronically only.

Delivery format. Specifies the format the PTFs are stored. Possible values are:

0	PTFs are delivered electronically as save files.
1	PTFs are delivered electronically as optical image files. The optical image files will contain PTFs and cover letters. The optical image files will be stored in the directory specified in the image directory field.

Delta level. Specifies the level of the system release.

Dependent address 2. An address field where the type of address is dependent on the address type field.

Address Type	**Dependent Address 2**
1 (Communications)	Adapter
2 (Storage)	Controller
3 (Workstation)	Adapter
4 (Auxiliary processor)	Auxiliary processor
5 (Library)	Library

Dependent address 3. An address field where the type of address is dependent on the address type field.

Address Type	**Dependent Address 3**
1 (Communications)	Port
2 (Storage)	Device
3 (Workstation)	Port
4 (Auxiliary processor)	Adapter
5 (Library)	Controller

Dependent address 4. An address field where the type of address is dependent on the address type field.

Address Type	Dependent Address 4
1 (Communications)	Channel
2 (Storage)	Reserved
3 (Workstation)	Device
4 (Auxiliary processor)	Port
5 (Library)	Device

Dependent address 5. An address field where the type of address is dependent on the address type field.

Address Type	Dependent Address 5
1 (Communications)	Reserved
2 (Storage)	Reserved
3 (Workstation)	Shared session
4 (Auxiliary processor)	Reserved
5 (Library)	Reserved

Device location. The physical location of the device.

Device location text. Text that describes the location of the device.

Device location text length. Length of text.

Device name. A name given to the device or feature.

Device type. The type of device located on the system.

Document reference message ID. Message that contains a description of reference material.

DSA bus number. Code further defining the electrical address of a resource.

EC number. Engineering change number.

Error code message identifier. Identifier of the message that describes the error log entry.

Error log identifier. Number of the error log.

Event date and time. Date and time event was added to the problem log entry.

Exit point of the PDP. A code that defines the procedure in the PDP that isolated the problem.

Fault symptom code. A code defining the symptom of the problem.

Feature. Feature of the device. This is set to blank if a feature is not applicable.

Feature length. Length of the feature field. Maximum length supported is 4.

File count. The number of series 4001 or 4002 keys that are concatenated to this key.

File library name. Name of the library that contains the file.

File member name. Name of the file member. This is *SAVF if the file is a save file. This is *NONE if the file has no members.

File name. The file name that was specified by the user program when the file was created, or the name of the device file used to create this file.

File number. The file number for this spooled file.

File type. The type of entry to process. The values are:

0	All entries
1	Spooled file entry
2	Data file entry

Filter event Defines if problem log should be filtered

0	Not set
1	Filter set

Filter group assigned. Name of the group in the filter to which the problem is assigned.

Filter library name. Library where the filter is located.

Filter name. Name of the filter.

Frame ID location. The physical location of the frame ID.

FRU code. A code that defines the FRU.

FRU count. Number of FRU entries that were returned by the Retrieve Problem Log API.

FRU description message ID. Message that describes this FRU.

Function. Name of the failing function.

Function length. Length of the function data.

Help desk or pager number. The help desk or pager number of the contact for the problem being reported. This number should include the area code, exchange numbers, and the extension.

Hierarchy. The function of the program where the problem occurred.

History count. The number of 6001 keys returned by the Retrieve Problem Log API.

History type. History entry type. The types are:

0	Problem entry closed
1	Problem entry opened
2	Service request received
3	Opened by an alert
4	Problem analyzed
5	Verification test ran
6	Recovery procedure ran
7	Prepared to report
8	Service request sent
9	Problem answered
10	Response sent
11	Reported by voice

12	Fixes transmitted
13	Change request submitted
14	Change request ended
15	Fix verified
16	Remote analysis
17	Remote verification ran
18	Remote recovery ran
19	Alert created
20	APAR created
21	APAR data collected
22	APAR data restored
23	APAR data deleted
24	Changed by CHGPRB
25	Deleted by DLTPRB
26	Problem occurred multiple times
27	Status changed
28	Status query sent
29	Problem automatically analyzed
30	Problem not automatically analyzed - SRC
31	Problem not automatically analyzed - SBMJOB
32	Automatic problem analysis failed
33	Problem sent automatically
34	Problem not sent automatically - SRC off
35	Problem not sent automatically - SBMJOB
36	Automatic problem notification failed
37	Problem analysis failed

Image directory. The path name of the directory where optical image files will be saved. For more information on specifying path names, refer to “Object naming rules” in “CL concepts and reference” in the CL reference information in the iSeries Information Center at http://www.iseries.ibm.com/infocenter. The following special value is accepted:

**DFT*	The optical image files are stored in /QIBM/UserData/OS/Service/ECS directory.

Image prefix. The prefix for the optical image file names. If multiple images are received under one order, the files will be uniquely identified by a numerical suffix on the image name. This field must be set to blanks if 1 (Image) is not specified for delivery format. The following special value is accepted:

**DFT*	No prefix will be added at the beginning of each optical image file name. The files will be named by the service provider.

Instruction. Instruction number where the error was detected.

I/O bus address. The bus number between the IOP and the device.

Isolation status. The status of the isolation attempt.

0	Not isolated, no FRUs added.
1	Completed successfully with isolation FRUs added.
2	Completed successfully, no problem found, point of failure FRUs added.
3	Unsuccessful, point of failure FRUs added.
4	Analysis not complete, point of failure FRUs added.
5	Analysis partially completed, partial FRU list added.

Job name. The name of the job that produced the spooled file.

Job number. The number of the job that produced this spooled file.

Key. Integer value that defines the key you are working with.

Key control for key 6. Defines the fields that will be processed.

1	Alert status
2	APAR library
4	Auto PAR
8	Auto Notify
16	From System
32	To System
64	Prepared For

Key control for key 8. Defines which field should be processed. Add the values to process multiple fields.

1	Use answer code assigned
2	Use answer code returned
4	Use problem number
8	Use problem branch number
16	Use problem country number

Key control for key 1004. Defines the EC (engineering change) number.

Key control for key 1005. Defines the fields that will be processed.

1	Type
2	Device
4	Product
8	Instruction
16	Hierarchy
32	Resource name
64	Error log identifier
128	Program

Key control for key 1006. Defines field used for reporting code:

1	Product name
2	Program name
4	Probe

Key control for key 1014. Defines which field should be processed. Add the values to process multiple fields.

1	Use count field
2	Use object length, object name and detecting product fields

Key control for key 5001. Defines contact data fields:

1	National Language Version (NLV)

2	Corporate name
4	Contact
8	Primary phone number
16	Help desk or pager number
32	Address
64	CCSID
128	Primary FAX contact phone number
256	Alternative FAX contact phone number

Key control for key 6001. Defines which fields to process:

0	Not use optional change request data
1	Use optional change request data

Key control for key 7001. Identifies which fields to process on a change operation. The control values are:

1	Status
2	Sent

To process multiple fields sum the values for the fields you want to change.

Key control for key 7002. Defines which fields to process on a change operation. The control values are:

1	Type
2	Option
4	Reorder
8	Delivery
16	Check
32	Delivery Format
64	Image directory
128	Image prefix

Key size. Defines the size of the key you are working with.

Machine type. A type of device.

Machine type length. Length of the machine type in bytes.

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.

Media type. This is a code that defines a media type.

1	Automatic selection. Auto selection implies that the system determines what device to use for fix distribution. This is required when the problem is PREPARED.
2	CD-ROM.

Message file name. The message file that contains the problem description. The library of the file must exist in the library list.

Message identifier. The identifier of a message that describes the problem.

Message library name. The library that contains the message file.

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.

Model. The model of the device type.

Model length. Size of the machine model field, maximum is 3.

Mode of analysis. Whether the problem was in message mode, which allows the user to analyze the problem, or the problem was analyzed by the System Licensed Internal Code (SLIC) and cannot be analyzed again. Valid values are as follows:

0	Message mode of analysis
1	SLIC mode of analysis

Modification level. Modification level of the object. *ONLY is a valid constant even though it is longer than the 2-byte maximum.

Modification level length. Length of the modification level in bytes. Maximum length supported is 2.

Module name. Component of a program or a program name.

Name. Product, microcode, application, or module name.

Name length. Length of the name of the object that detected the error.

Name of contact. The name of a person to contact within the corporation.

National language version. A code that defines the national language version in which the cover letter is supplied. The values are defined in the globalization topic.

Network address. Defines the address of a network node. These formats are used:

APPN	• Network ID • Control point name • Reserved
Internet (in dotted form)	• Address • Reserved
**IBMSRV*	• Network ID (must be blank) • Control point name (must be '*IBMSRV') • Reserved

The reserved fields must be blank.

Number of times analyzed. The number of times the problem was analyzed. The value must be greater than 0 and should be incremented each time the problem is analyzed.

Object name. Name of the object that detected the error.

Option. Defines if only the PTFs listed will be ordered or the PTFs and its requisites.

0	PTF with no requisites
1	PTF and requisites

Part location. A textual description of where the part is located.

Part number array. List of up to six part numbers, 12 characters in length. Unused elements of the array should be blank.

Path name. Path to the failing software.

Path name length. Length of the path name data.

PDP. Name of the problem determination procedure (PDP) module used to isolate the error.

Pointer to the text. Address of the text.

Port. Code defining where a device is attached to a device driver.

Postal code. The postal or zip code of the postal address.

Primary FAX number. The primary FAX number to receive problem-related data.

Primary function group. The load ID of the program.

Primary phone number. The phone number of the primary contact for the problem being reported.

Probability of fix. The probability of this FRU resolving the problem.

Probe. Identifier for a problem found in a program.

Problem branch number. A number assigned by the support system. The problem branch number field is typically the problem management branch number used by *IBMSRV.

Problem category. Defines how a problem should be processed.

0	REPORT-Designates a set of problem log entries that can be reported. This includes all problems except for LOGONLY problems.
1	CRITICAL-Designates a set of problem log entries that have been created from a critical message. These problems should be handled immediately.
2	LOGONLY-Designates a set of problems that are log-only. These problems cannot be reported.
3	ALL-All program log entries are displayed

Problem country number. A number assigned by the support system. The problem country number field is typically the problem management country number used by *IBMSRV.

Problem log identifier. A unique identifier based on date and time, network type and network address. The values are:

Number	On a create operation, this key defines whether the problem is being created for a local or remote problem. A constant of ″*LOCAL″ is used to identify the problem as a local one. The problem log ID is provided in key 1 (Problem log ID) when a remote problem is being created.

Network type	Network type *A* APPN address
Network address	Identifies the network in which the server resides. The format is: • 8 characters for the network ID • 8 characters for the control point name • 4 characters reserved (must be blank)

Problem number. A number assigned by the support system. The problem number field is typically the problem management number used by *IBMSRV.

Problem severity. The impact of the problem on the system. The values are:

1	High
2	Medium
3	Low
4	None

Problem status. Defines the current status of the problem. The values are:

0	*OPENED status
1	*READY status
2	*SENT status
3	*ANSWERED status
4	*VERIFIED status
5	*PREPARED status
6	*CLOSED status

Problem type. Defines the type of problem the system is processing. The values are:

1	Machine-detected problem
2	User-perceived hardware or software problem
3	PTF orders
4	User-perceived remote problem
5	Application-detected problem
6	Client machine-detected problem
7	Client user-detected problem
8	User-created general problem

Product ID. Name of the product.

Product ID length. Length of the product ID data. The maximum length is 7 except for key 1013 where the maximum is 64. *ONLY*PRODUCT** is a valid constant even though it is longer than the 7-byte maximum.

Program. Name of the failing program.

Program length. Length of the program data.

PTF count. Number of PTF entries retrieved.

PTF ID. A program fix identifier.

PTF ID length. Length of the program fix identifier. Maximum length is 7.

PTF image. Identifies whether or not the PTF was downloaded as an optical image file on the system. Possible values are:

0	The PTF was not downloaded as an optical image file.
1	The PTF was downloaded on the system as an optical image file.

PTF order type. Defines if the PTF and its cover letter will be ordered or only the cover letter.

0	PTF and cover letter
1	Cover letter only

PTF status. Identifies whether the PTF has been requested from a remote system. Requested implies that the PTF order was sent and the PTF is needed by your system.

0	PTF not requested
1	PTF requested

Query status. Defines how the client service information is to be processed.

0	Field not defined
1	Service representative opened the problem
2	Service representative has been dispatched
3	Problem closed
4	Problem closed and service representative has been dispatched

Rack serial number. Serial number of the rack.

Rack serial number length. Length of the serial number of the rack.

Recovery status. The status of the recovery attempt.

0	Recovery status not available
1	Recovery status available
2	Fix verified
3	Recovery failed

Reference code. Index into a reference code translatable table.

Reference code description data. Data defining the error.

Reference code description file name. File that contains the reference code description.

Reference code description library name. Library that contains the reference code description.

Reference code description message. Message identifier that describes the problem.

Reference code translate table identifier. Name of the table that contains a description of the problem.

Reference code translate table library. Library that contains the reference code translate table.

Release level. Release level of the object. *ONLY is a valid constant even though it is longer than the 2-byte maximum.

Release level length. Maximum length supported is 2.

Reorder. Defines if a PTF that is already on the system, but which does not have a save file, will be reordered. Typically a PTF will not be ordered if it has been loaded or installed. This option overrides normal operation but if a save file exists for the PTF it will not be reordered.

0	Do not reorder a PTF that is available on the system.
1	Order a PTF for which there is no save file for that PTF exists on the system.

Replacement text. Defining the configuration error.

Replacement text length. Length of the data in bytes.

Reserved. Space added to ensure correct boundary alignment. This field must be blank.

Resource name. Name of the resource.

Screen identifier. The identifier of a screen to be displayed to assist in solving a problem.

Secondary function group. Program option.

Sense bytes. Sense bytes that pertain to the problem.

Sent. Defines if the PTF has been sent from the remote system in response to a PTF order or problem report.

0	PTF not sent
1	PTF sent

Serial number. Manufacturing sequence number or designation.

Serial number length. Maximum length supported is 7.

Symptom. An encoded string that represents the problem description. Typically, this field contains EBCDIC uppercase alphabetic, numeric, and limited special characters. Contact your service representative for data restrictions. This field is considered an user-defined field and no translation or alteration of the contents are made. The first character position of the field cannot be blank.

Text count. A count of 3001 entries returned by the Retrieve Problem Log API of entries returned. If no entries are found, 0 is returned.

Text length. Length of the data in bytes.

Text type. A code that defines the type of text to process. The values are:

0	All text, used on key 3000 (text entry) only to retrieve all entries.
1	80-character title, limit to one entry
2	Long problem description
3	Problem status
4	Private notes
5	Associated problem data

Transport type. The type of connection from the central electronics complex (CEC) to the board's user-assigned value for this SPD bus.

Type of hardware. Machine, device, feature, or component type.

Unit address. Code defining the electrical address of a resource.

User assigned. The user profile of the person assigned to this problem. The value is blank if not assigned.

User ID. User ID of the job making the entry.

Verification status. Defines the status of the recovery attempt.

0	Not available
1	Available
2	Fixed
3	Failed

Version. Release level of the product. *ONLY is a valid constant even though it is longer than the 2-byte maximum.

Version length. Length of the version data. Maximum length supported is 2 except for key 1013 where the maximum is 64.

Volume ID. Identifier of the media that is failing.

Top | "Problem Management APIs," on page 1 | APIs by category

Appendix. Notices

Trademarks

DRDA
eServer
GDDM
IBM
Integrated Language Environment
Intelligent Printer Data Stream
IPDS
iSeries
Lotus Notes
MVS
Netfinity
Net.Data
NetView
Notes
OfficeVision
Operating System/2
Operating System/400
OS/2
OS/400
PartnerWorld
PowerPC
PrintManager
Print Services Facility
RISC System/6000
RPG/400
RS/6000
SAA
SecureWay
System/36
System/370
System/38
System/390
VisualAge
WebSphere
xSeries

Microsoft, Windows, Windows NT, and the Windows logo are trademarks of Microsoft Corporation in the United States, other countries, or both.

Java and all Java-based trademarks are trademarks of Sun Microsystems, Inc. in the United States, other countries, or both.

UNIX is a registered trademark of The Open Group in the United States and other countries.

Other company, product, and service names may be trademarks or service marks of others.

Terms and conditions for downloading and printing publications

Permissions for the use of the information you have selected for download are granted subject to the following terms and conditions and your indication of acceptance thereof.

Personal Use: You may reproduce this information for your personal, noncommercial use provided that all proprietary notices are preserved. You may not distribute, display or make derivative works of this information, or any portion thereof, without the express consent of IBM(R).

Code disclaimer information

Printed in USA