IBM Systems - iSeries
Network Management APIs

Version 5 Release 4
## Contents

**Network Management APIs** .............................................. 1

- APIs .................................................................................. 1
- APPN Topology Information APIs ........................................ 1
  - Local and Network Topology Updates ................................. 2
  - APPM Network Topology Updates ..................................... 2
  - APPN Local Topology Updates ......................................... 2
  - Adjacent Subnetworks ..................................................... 3
- Deregister APPN Topology Information (QNMDRGTI) API ............ 3
  - Authorities and Locks .................................................... 3
  - Required Parameter Group ............................................. 3
  - Error Messages ............................................................ 4
- Register APPN Topology Information (QNRGTM) API .................. 4
  - Authorities and Locks .................................................... 5
  - Required Parameter Group ............................................. 5
  - Format of the Generated List ......................................... 7
  - Field Descriptions ....................................................... 9
  - Format of Node Attributes Field ...................................... 13
  - Format of Transmission Group Flags Field ......................... 14
  - Format of Transmission Group Characteristics .................... 15
  - Error Messages ............................................................ 16
- SNA/Management Services Transport APIs ............................. 16
  - Using the SNA/Management Services Transport APIs .......... 18
  - Entry Format .................................................................. 18
  - Field Descriptions ....................................................... 18
  - Data ........................................................................... 20
  - Routing ....................................................................... 21
- Change Mode Name (QNMCCHGMN) API .................................. 21
  - Authorities and Locks .................................................... 21
  - Required Parameter Group ............................................. 22
  - Error Messages ............................................................ 22
- Deregister Application (QNMDRGAP) API ................................. 22
  - Authorities and Locks .................................................... 23
  - Required Parameter Group ............................................. 23
  - Error Messages ............................................................ 23
- End Application (QNMENDAP) API ......................................... 23
  - Authorities and Locks .................................................... 24
  - Required Parameter Group ............................................. 24
  - Error Messages ............................................................ 24
- Receive Data (QNRCDVDT) API ............................................ 24
  - Authorities and Locks .................................................... 24
  - Required Parameter Group ............................................. 25
  - Error Messages ............................................................ 26
- Receive Operation Completion (QNMRVOC) API ........................ 26
  - Authorities and Locks .................................................... 27
  - Required Parameter Group ............................................. 27
  - Error Messages ............................................................ 27
- Register Application (QNRREGAP) API ................................... 28
  - Authorities and Locks .................................................... 28
  - Required Parameter Group ............................................. 28
  - Error Messages ............................................................ 29
- Retrieve Mode Name (QNMRVMN) API .................................... 29
  - Authorities and Locks .................................................... 29
  - Required Parameter Group ............................................. 29

**Error Messages** ................................................................ 30

- Send Error (QNMSNDR) API ............................................... 30
  - Authorities and Locks .................................................... 30
  - Required Parameter Group ............................................. 30
  - Error Messages ............................................................ 31
- Send Reply (QNMSNDRF) API .............................................. 31
  - Required Parameter Group ............................................. 32
  - Error Messages ............................................................ 33
- Send Request (QNMSND) API ............................................. 33
  - Authorities and Locks .................................................... 34
  - Required Parameter Group ............................................. 34
  - Error Messages ............................................................ 35
- Start Application (QNMRSTRAP) API ..................................... 35
  - Authorities and Locks .................................................... 36
  - Required Parameter Group ............................................. 36
  - Error Messages ............................................................ 36
- Alert APIs .......................................................................... 37
  - Generate Alert (QALGENA) API ....................................... 37
  - Authorities and Locks .................................................... 37
  - Required Parameter Group ............................................. 37
  - Error Handling ............................................................. 39
  - Error Messages ............................................................ 39
- Retrieve Alert (QALRTVA) API ............................................ 39
  - Authorities and Locks .................................................... 39
  - Required Parameter Group ............................................. 40
  - ALRT0100 Format .......................................................... 40
  - ALRT0100 Format .......................................................... 41
  - Alert Notification Record ................................................. 41
  - Field Descriptions ........................................................ 41
  - Error Messages ............................................................ 43
- Send Alert (QALSND) API ................................................... 43
  - Authorities and Locks .................................................... 43
  - Required Parameter Group ............................................. 43
  - Error Messages ............................................................ 44
- Node List API ..................................................................... 44
  - Registered Filter APIs .................................................... 48
  - Deregister Filter Notifications (QNMDFGN) API ................. 49
    - Authorities and Locks ................................................ 49
    - Required Parameter Group ......................................... 49
    - Error Messages ........................................................ 50
  - Register Filter Notifications (QNMDFGN) API ..................... 50
    - Authorities and Locks ................................................ 50
    - Required Parameter Group ......................................... 51
    - Format of Registered Filter Data Queue ......................... 51
    - Notification ............................................................. 51
    - Alert Filter .............................................................. 51
    - Problem Log Filter .................................................... 52

© Copyright IBM Corp. 1998, 2006
Network Management APIs

Network management is the process of planning, organizing, and controlling a communications-oriented system. It provides you with the capability to manage one or more nodes from another node. The network management APIs handle alertable messages, work with problem logs, and the SNA/Management Services Transport.

The network management APIs are grouped as follows:

- "APPN Topology Information APIs"
- "SNA/Management Services Transport APIs" on page 16
- "Alert APIs" on page 37
- "Node List API" on page 44
- "Registered Filter APIs" on page 48
- "Change Request Management APIs" on page 56

APIs by category

APIs

These are the APIs for this category.

APPN Topology Information APIs

The APPN(R) topology information APIs are:

- "Deregister APPN Topology Information (QNMDRGTI) API” on page 3 (QNMDRGTI) causes the queue associated with the specified queue handle to be deregistered for APPN topology information.
- "Register APPN Topology Information (QNMRGTI) API” on page 4 (QNMRGTI) causes the requested APPN topology information to be reported.

APPN topology information APIs allow an application to obtain information about the current APPN topology, and to register and deregister for information about ongoing updates to the topology. Current topology information is an option provided by the Register for APPN Topology Information (QNMRGTI) API. When this option is requested, the current APPN topology is reported to a user space specified by the application running the API. This API also provides topology update options that allow the application to register a queue to receive information about specific types of APPN topology updates. Topology updates are reported asynchronously to the specified queue as they occur in the network.

A queue remains registered for topology updates until one of the following occurs:

- The queue is deregistered by the application using the Deregister APPN Topology Information (QNMDRGTI) API.
- An error is encountered enqueuing topology updates, forcing automatic deregistration of the queue by the system.
- The application’s job is ended causing registration to be cleaned up.
- An IPL is performed causing registration to be cleaned up.

One application in each job on the system may register one queue for topology updates. Multiple queues may not be registered for topology updates within the same job.

The specific types of topology updates that an application may register to receive are:
• Local end node (*EN) updates
• Local virtual node (*VN) updates
• Local network node (*NN) updates
• Network network node (*NN) updates
• Network virtual node (*VN) updates

The queue and user space objects specified on the APPN topology information APIs must be managed entirely by the application; for example, the application must create, delete and maintain the objects itself, using the APIs for those objects. The application is responsible for any error handling should these objects become damaged or deleted.

If an error occurs while reporting the current topology to the specified user space, an error is returned through the API. If an error occurs while enqueuing ongoing topology updates to a registered queue, the resulting error messages are sent to the job log, followed by diagnostic message CPD91C9, and the queue is automatically deregistered by the system. If automatic handling of this diagnostic error message is necessary, an application could periodically scan the job log for this message using the List Job Log (QMHLJOBL) API and take appropriate action.

After the current topology has been requested using the QNMRTG1 API, the data returned in the user space may be retrieved using the Retrieve User Space (QUSRTVUS) API. If a data queue is registered for topology updates using the QNMRTG1 API, topology update records may be retrieved out of a data queue using the Receive Data Queue (QRCVDTAQ) API. If a user queue is registered (rather than a data queue), the application must use the dequeue (DEQ) MI instruction to retrieve queue records. The first 10 characters of each queue entry contains the value *APPNtop so that the application can distinguish these records from others on the queue. This allows a queue to be used for multiple purposes.

Local and Network Topology Updates

Topology updates can be separated into two classes:
• Network topology updates
• Local topology updates

Local topology updates can be reported on an end node or network node system, but network topology updates can be reported only on a network node system.

APPN Network Topology Updates

An APPN subnetwork consists of nodes having a common network ID and the links connecting those nodes. APPN network topology identifies the following in an APPN subnetwork:
• All network nodes and virtual nodes in the subnetwork
• Transmission groups interconnecting network nodes and virtual nodes in the subnetwork
• Transmission groups from network nodes in the subnetwork to nodes in adjacent subnetworks

APPN network nodes exchange network topology updates in a subnetwork through topology database updates (TDUs). Therefore, only network nodes can report network topology updates. See System Network Architecture Formats for details about TDUs.

APPN Local Topology Updates

The local topology for an APPN node consists of the following:
• The local node
• Adjacent nodes (network nodes, end nodes, or virtual nodes to which the local node has a direct connection)
• Transmission groups from the local node to adjacent nodes
Both end nodes and network nodes can report local topology updates.

**Adjacent Subnetworks**

Network nodes in separate subnetworks may be connected by an intersubnetwork transmission group; that is, a group of links between directly attached nodes of 2 or more subnetworks appearing as a single logical link for routing messages. In this case, the network node at each end point of the transmission group is present only in the partner network node’s local topology, not in its network topology. For example, consider two network nodes with different network IDs in separate subnetworks:

- A network node with the following control point name and network ID: CPNAME=NN1, NETWORK ID=A
- A network node with the following control point name and network ID: CPNAME=NN2, NETWORK ID=B

When NN1 and NN2 are connected by an intersubnetwork transmission group, NN2 is present only in the NN1 local topology; it is not present in the NN1 network topology or other nodes in network A. This is because TDUs for NN2 are not exchanged in network A.

**Deregister APPN Topology Information (QNMDRGTI) API**

Required Parameter Group:

<table>
<thead>
<tr>
<th></th>
<th>Queue handle</th>
<th>Input</th>
<th>Binary(4)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Error code</td>
<td>I/O</td>
<td>Char(*)</td>
</tr>
</tbody>
</table>

Default Public Authority: *USE
Threadsafe: No

The Deregister APPN Topology Information (QNMDRGTI) API causes the queue associated with the specified queue handle to be deregistered for APPN topology information. When a queue is deregistered, future topology updates are not reported to the queue.

The queue handle specified by this API must match the queue handle that was returned when the queue was registered with the Register APPN Topology Information (QNMRGTI) API.

**Authorities and Locks**

None.

**Required Parameter Group**

**Queue handle**

INPUT; BINARY(4)

A variable that represents a registered queue. This value was returned by the QNMRGTI API when the queue was registered.

**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**

<table>
<thead>
<tr>
<th>Message ID</th>
<th>Error Message Text</th>
</tr>
</thead>
<tbody>
<tr>
<td>CPF24B4 E</td>
<td>Severe error while addressing parameter list.</td>
</tr>
<tr>
<td>CPF3C90 E</td>
<td>Literal value cannot be changed.</td>
</tr>
<tr>
<td>CPF3CF1 E</td>
<td>Error code parameter not valid.</td>
</tr>
<tr>
<td>CPF91C3 E</td>
<td>Internal processing error.</td>
</tr>
<tr>
<td>CPF91C6 E</td>
<td>Queue handle &amp;1 not valid.</td>
</tr>
<tr>
<td>CPF9872 E</td>
<td>Program or service program &amp;1 in library &amp;2 ended. Reason code &amp;3.</td>
</tr>
</tbody>
</table>

API introduced: V2R2

---

**Register APPN Topology Information (QNMRGTI) API**

Required Parameter Group:

<table>
<thead>
<tr>
<th></th>
<th>Queue handle</th>
<th>Options</th>
<th>Number of options</th>
<th>Format</th>
<th>Qualified user space name</th>
<th>Qualified queue name</th>
<th>Queue type</th>
<th>Replace registration</th>
<th>Error code</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Output</td>
<td>Input</td>
<td>Input</td>
<td>Input</td>
<td>Input</td>
<td>Char(*)</td>
<td>I/O</td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>Binary(4)</td>
<td>Array of Char(10)</td>
<td>Binary(4)</td>
<td>Char(8)</td>
<td>Char(20)</td>
<td>Char(20)</td>
<td>Char(10)</td>
<td>Char(10)</td>
<td>Char(*)</td>
</tr>
</tbody>
</table>

Default Public Authority: *USE
Threadsafe: No

The Register APPN Topology Information (QNMRGTI) API causes the requested APPN topology information to be reported. The application calling this API may request the current copy of the entire database (to be reported to the specified user space), or may register for information about particular types of ongoing updates to the topology (to be reported to the registered queue), or both.

A queue handle is returned by this program when a queue is successfully registered for topology updates. The queue handle identifies a registered queue, and must be used when the Deregister APPN Topology Information (QNMDRGTI) API is called. The queue handle is unique to a specific job.

When a queue is registered for ongoing topology updates, the specified types of updates for which the queue is registered will be asynchronously enqueued on an ongoing basis. If current topology is also requested on the application calling this API, the current topology is reported to the user space before topology updates are reported to the registered queue.

The QNMRGTI API may be called to request the current topology only (without updates), updates only (without current topology), or both current topology and updates. This is determined by the options specified in the options parameter.

If an application program calling the API requests current topology, and the complete topology data cannot be returned in the user space, an error is returned. In this situation, the user space header contains a P in the information status field indicating partial but accurate data. Even if a user space error such as this occurs, any queue registered on the API call will remain registered.
Authorities and Locks

User Space Authority
   *CHANGE

User Space Library Authority
   *EXECUTE

Queue Authority
   *CHANGE

Queue Library Authority
   *USE

User Space Lock
   *EXCLRD

Required Parameter Group

Queue handle
   OUTPUT; BINARY(4)
   A variable that uniquely identifies the registered queue within the job. This value is returned when a queue is registered. When the *CURRENT value is the only option specified in the options parameter, 0 is returned.

Options
   INPUT; Array OF CHAR(10)
   An array structure containing options specifying the topology information to be reported. These options only apply to topology updates. All deletions are reported regardless of the type of updating information you want.

   One or more of the following values must be specified:

   *CURRENT    Report the current copy of the entire topology database.
   *LOCALEN    Register for local topology updates pertaining to adjacent end nodes. Local topology consists of the local node, adjacent nodes, and links to adjacent nodes. See Local and Network Topology Updates for more details.
   *LOCALNN    Register for local topology updates pertaining to adjacent network nodes. This option must be used if updates pertaining to adjacent network nodes in a disconnected subnetwork, such as network nodes having a different network ID, are required.
   *LOCALVN    Register for local topology updates pertaining to adjacent virtual nodes. An APPN virtual node represents a connection network (for example, an attached token-ring network). For more information, see the book Distributed Data Management.
   *NETNN      Register for network topology updates pertaining to network nodes.
   *NETVN      Register for network topology updates pertaining to virtual nodes.

Number of options
   INPUT; BINARY(4)
   The number of options specified in the options parameter. Valid values are 1 through 6.

Format
   INPUT; CHAR(8)
   The content and format of the topology information reported. The valid values are:

   APPN0100    The basic APPN topology information format. See APPN0100 Format (page [3]) for a description of this format.
   APPN0200    The basic APPN topology information format. See APPN0200 Format (page [3]) for a description of this format.
Qualified user space name

INPUT; CHAR(20)

The user space that is to receive current topology information. This parameter is ignored when the *CURRENT value is not specified on the options parameter. The first 10 characters specify the user space name, and the last 10 characters specify the library name.

The following special values are supported for the library name:

*LIBL The library list.
*CURLIB The job’s current library.

Qualified queue name

INPUT; CHAR(20)

The queue that is to receive requested topology information. This parameter is ignored when the *CURRENT value is the only option specified on the options parameter. The first 10 characters specify the queue object name, and the last 10 characters specify the library name.

The following special values are supported for the library name:

*LIBL The library list.
*CURLIB The job’s current library.

When special values are used for the library name, the actual name will be substituted when the API is called. The actual name will be used in future references to the object when topology updates are enqueued.

The following considerations apply to the queue specified on the API call:

• The queue object must exist when this API is called.
• There is no restriction that prevents applications in separate jobs from registering the same queue object. Therefore, each application should ensure it registers a unique queue to prevent duplicate topology updates from being reported to the same queue. For example, specifying a queue in library QTEMP ensures the queue is not being used by other jobs on the system.
• The maximum entry length of the queue must be at least 128 bytes.
• The sequence of the queue must be *FIFO.
• A data queue defined with FORCE(*YES) is allowed, but is discouraged due to degraded performance.
• A data queue defined with SENDERID is allowed, but the application is responsible for ensuring sufficient record length to handle the additional data on queue elements.

Queue type

INPUT; CHAR(10)

The type of queue object. This parameter is ignored when *CURRENT is the only value specified on the options parameter.

Otherwise, one of the following values must be specified:

*DTAQ Data queue
*USRQ User queue

Replacement Registration

INPUT; CHAR(10)

Whether this registration should replace a previous registration that has the same qualified queue name and type. This parameter is ignored when *CURRENT is the only value specified on the
options parameter.

*YES This registration replaces any previous registration with the same qualified queue name and type. The options specified on this registration replace options from any previous registration.

*NO This registration does not replace any existing registration with the same qualified queue name and type. The existing registration is not changed.

Error code

I/O; CHAR(*)

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

Format of the Generated List

The user space is used to report current topology information and consists of:

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

For details about the user area and generic header, see User Space Format for List APIs. For details about the other items, see the following sections. For a detailed description of each field in the information returned, see “Field Descriptions” on page 9.

Input Parameter Section

<table>
<thead>
<tr>
<th>Offset</th>
<th>Dec</th>
<th>Hex</th>
<th>Type</th>
<th>Field</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>CHAR(10)</td>
<td>User space name specified</td>
</tr>
<tr>
<td>10</td>
<td>A</td>
<td></td>
<td>CHAR(10)</td>
<td>User space library name specified</td>
</tr>
<tr>
<td>20</td>
<td>14</td>
<td></td>
<td>CHAR(8)</td>
<td>Format name specified</td>
</tr>
</tbody>
</table>

Header Section

<table>
<thead>
<tr>
<th>Offset</th>
<th>Dec</th>
<th>Hex</th>
<th>Type</th>
<th>Field</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>CHAR(10)</td>
<td>User space name used</td>
</tr>
<tr>
<td>10</td>
<td>A</td>
<td></td>
<td>CHAR(10)</td>
<td>User space library name used</td>
</tr>
<tr>
<td>20</td>
<td>14</td>
<td></td>
<td>BINARY(4)</td>
<td>Number of nodes reported</td>
</tr>
<tr>
<td>24</td>
<td>18</td>
<td></td>
<td>CHAR(8)</td>
<td>Timestamp of list data</td>
</tr>
<tr>
<td>32</td>
<td>20</td>
<td></td>
<td>CHAR(8)</td>
<td>Timestamp when local node initialized</td>
</tr>
<tr>
<td>40</td>
<td>28</td>
<td></td>
<td>BINARY(4)</td>
<td>Number of end nodes reported</td>
</tr>
<tr>
<td>44</td>
<td>2C</td>
<td></td>
<td>BINARY(4)</td>
<td>Number of transmission groups reported for all end nodes</td>
</tr>
<tr>
<td>48</td>
<td>30</td>
<td></td>
<td>BINARY(4)</td>
<td>Number of transmission groups reported for all network and virtual nodes</td>
</tr>
<tr>
<td>52</td>
<td>34</td>
<td></td>
<td>CHAR(8)</td>
<td>Local node network ID</td>
</tr>
<tr>
<td>60</td>
<td>3C</td>
<td></td>
<td>CHAR(8)</td>
<td>Local node control point name</td>
</tr>
<tr>
<td>68</td>
<td>44</td>
<td></td>
<td>BINARY(4)</td>
<td>Local topology database flow reduction sequence number</td>
</tr>
</tbody>
</table>
### APPN0100 Format

The format of topology entry data is the same for queue entries reported on the registered queue for topology updates and deletions as for list entries reported to the user space for the current topology.

For topology entries reported to the user space, the entry ID field contains the value that indicates a current topology entry.

For topology entries reported to the registered queue, the entry ID field contains a value that identifies the type of notification reported:

- Topology entry updated
- Topology entry deleted
- Topology database deleted

The format of the topology entry is described below. See “Field Descriptions” on page 9 for descriptions of the fields in this format.

<table>
<thead>
<tr>
<th>Offset</th>
<th>Dec</th>
<th>Hex</th>
<th>Type</th>
<th>Field</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td>CHAR(10)</td>
<td>Entry type</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>A</td>
<td>CHAR(2)</td>
<td>Entry ID</td>
</tr>
<tr>
<td>10</td>
<td>C</td>
<td>12</td>
<td>CHAR(8)</td>
<td>Timestamp</td>
</tr>
<tr>
<td>20</td>
<td>14</td>
<td>20</td>
<td>CHAR(8)</td>
<td>Node network ID</td>
</tr>
<tr>
<td>28</td>
<td>1C</td>
<td>28</td>
<td>CHAR(8)</td>
<td>Node control point name</td>
</tr>
<tr>
<td>36</td>
<td>24</td>
<td>36</td>
<td>BINARY(4)</td>
<td>Number of associated transmission group entries</td>
</tr>
<tr>
<td>40</td>
<td>28</td>
<td>40</td>
<td>CHAR(1)</td>
<td>Node data valid indicator</td>
</tr>
<tr>
<td>41</td>
<td>29</td>
<td>41</td>
<td>CHAR(3)</td>
<td>Node type</td>
</tr>
<tr>
<td>44</td>
<td>2C</td>
<td>44</td>
<td>CHAR(8)</td>
<td>Node attributes</td>
</tr>
<tr>
<td>52</td>
<td>34</td>
<td>52</td>
<td>CHAR(8)</td>
<td>Transmission group destination network ID</td>
</tr>
<tr>
<td>60</td>
<td>3C</td>
<td>60</td>
<td>CHAR(8)</td>
<td>Transmission group destination control point name</td>
</tr>
<tr>
<td>68</td>
<td>44</td>
<td>68</td>
<td>BINARY(4)</td>
<td>Transmission group number</td>
</tr>
<tr>
<td>72</td>
<td>48</td>
<td>72</td>
<td>CHAR(20)</td>
<td>Transmission group characteristics</td>
</tr>
<tr>
<td>92</td>
<td>5C</td>
<td>92</td>
<td>BINARY(4)</td>
<td>Length of DLC signaling information</td>
</tr>
<tr>
<td>96</td>
<td>60</td>
<td>96</td>
<td>CHAR(16)</td>
<td>DLC signaling information</td>
</tr>
</tbody>
</table>
### APPN0200 Format

See [“Field Descriptions”](#) for descriptions of the fields in this format.

<table>
<thead>
<tr>
<th>Offset</th>
<th>Dec</th>
<th>Hex</th>
<th>Type</th>
<th>Field</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>CHAR(10)</td>
<td>Entry type</td>
</tr>
<tr>
<td>10</td>
<td>10</td>
<td>A</td>
<td>CHAR(2)</td>
<td>Entry ID</td>
</tr>
<tr>
<td>12</td>
<td>12</td>
<td>C</td>
<td>BINARY(4)</td>
<td>Node entry time left</td>
</tr>
<tr>
<td>16</td>
<td>16</td>
<td>10</td>
<td>BINARY(4)</td>
<td>Node flow reduction sequence number</td>
</tr>
<tr>
<td>20</td>
<td>20</td>
<td>14</td>
<td>BINARY(2)</td>
<td>Length of node network-qualified name</td>
</tr>
<tr>
<td>22</td>
<td>22</td>
<td>16</td>
<td>CHAR(17)</td>
<td>Node network-qualified control point name</td>
</tr>
<tr>
<td>39</td>
<td>39</td>
<td>27</td>
<td>CHAR(1)</td>
<td>Reserved</td>
</tr>
<tr>
<td>40</td>
<td>40</td>
<td>28</td>
<td>BINARY(4)</td>
<td>Node type enumeration</td>
</tr>
<tr>
<td>44</td>
<td>44</td>
<td>2C</td>
<td>CHAR(8)</td>
<td>Node attributes</td>
</tr>
<tr>
<td>52</td>
<td>52</td>
<td>34</td>
<td>BINARY(4)</td>
<td>Number of associated transmission group entries</td>
</tr>
<tr>
<td>56</td>
<td>56</td>
<td>38</td>
<td>CHAR(4)</td>
<td>Transmission group number</td>
</tr>
<tr>
<td>60</td>
<td>60</td>
<td>3C</td>
<td>BINARY(2)</td>
<td>Length of transmission group destination network-qualified name</td>
</tr>
<tr>
<td>62</td>
<td>62</td>
<td>3E</td>
<td>CHAR(17)</td>
<td>Transmission group destination network-qualified control point name</td>
</tr>
<tr>
<td>79</td>
<td>79</td>
<td>4F</td>
<td>CHAR(1)</td>
<td>Transmission group flags</td>
</tr>
<tr>
<td>80</td>
<td>80</td>
<td>50</td>
<td>CHAR(20)</td>
<td>Transmission group characteristics</td>
</tr>
<tr>
<td>100</td>
<td>100</td>
<td>64</td>
<td>BINARY(4)</td>
<td>Transmission group entry time left</td>
</tr>
<tr>
<td>104</td>
<td>104</td>
<td>68</td>
<td>BINARY(4)</td>
<td>Transmission group flow reduction sequence number</td>
</tr>
<tr>
<td>108</td>
<td>108</td>
<td>6C</td>
<td>BINARY(2)</td>
<td>Length of DLC signaling information</td>
</tr>
<tr>
<td>110</td>
<td>110</td>
<td>6E</td>
<td>CHAR(8)</td>
<td>DLC signaling information</td>
</tr>
<tr>
<td>118</td>
<td>118</td>
<td>76</td>
<td>CHAR(10)</td>
<td>Controller description object name</td>
</tr>
</tbody>
</table>

### Field Descriptions

**Controller description object name.** The name of the controller description object for the transmission group. This field is valid only when the number of associated transmission group entries field is not zero. The object name is only available for transmission group entries associated with the local node, and only when the object currently exists on the local system. When the object name is not available, this field contains blanks.
**DLC signaling information.** The data link control (DLC) signaling information related to the link connection network. For token ring, the first 6 bytes is the MAC address, and the seventh byte is the link layer service access point address. This field is valid only when the number of associated transmission group entries field is not zero.

**Entry ID.** The type of topology information. The possible values are listed below. Value 00 is always used in topology entries reported to only the user space.

The other three values are used for topology entries reported to the registered queue.

- **00** The entry reported is present in the current APPN topology database.
- **01** The entry reported was updated in the APPN topology database.
- **02** The entry reported was deleted from the APPN topology database. APPN performs cleanup of the APPN topology database once every 24 hours. This cleanup may cause multiple entries to be removed from the APPN topology database.
- **03** The current APPN topology database was deleted and reinitialized with the node entry for the local system. The node entry for the local system is reported on the entry. Any additions to the newly initialized topology database will follow in subsequent queue entries. This indication may be reported, for example, when the node type parameter changes as a result of the Change Network Attributes (CHGNETA) command, causing the APPN topology database to be deleted.

**Entry type.** The type of queue entry.

This field can contain the following special value:

- **APPNTO** The entry contains APPN topology information.

**Format name specified.** The format name specified to the API.

**Length of DLC signaling information.** The length of the data link control (DLC) signaling information. This field is valid only when the number of associated transmission group entries field is not zero. The value is 7 for token ring (only DLC currently allowed), or 0 when there is no DLC signaling information available.

**Length of local node network-qualified control point name.** The length of the node network-qualified control point name. Valid range is 3-17.

**Length of node network-qualified name.** The length of the node network-qualified name.

**Length of transmission group destination network-qualified name.** The length of the transmission group destination network-qualified control point name. When the number of associated transmission group entries is zero, this value is zero. When the number of associated transmission group entries is not zero, valid values range from 3 through 17.

**Local node control point name.** The control point name for the local node.

**Local node network ID.** The network ID for the local node.

**Local node network-qualified control point name.** The network-qualified control point name for the local node, in the format NETID.CPNAME.

**Local node type.** The APPN node type of the local node. The valid values are:

- **EN** APPN end node
- **NN** APPN network node
*VN  APPN virtual node

Local node type enumeration. A number representing the APPN node type of the node. The valid values are:

0  Node data not valid (this may occur when an entry only contains transmission groups owned by this node).
1  APPN network node
2  APPN end node
3  APPN virtual node

Local topology database flow reduction sequence number. The flow reduction sequence number (FRSN) incremented each time the local network node sends a topology database update. This field is valid only when the local node type is a network node.

Node attributes. The attributes of the node. See “Format of Node Attributes Field” on page 13 for the structure. For format APPN0100, the data in this field is valid only when the node data valid indicator field is Y. For format APPN0200, the data in this field is valid only when the node type enumeration is not 0.

Node control point name. The control point name for the node.

Node data valid indicator. Whether values contained in the format APPN0100 node type and attributes fields are valid. The valid values are:

Y  The values contained in the node type and the node attributes fields are valid.
N  The values contained in the node type and the node attributes fields are not valid. This may occur when an entry only contains associated transmission groups owned by this node.

Node entry time left. The number of days left before the node entry is deleted from the topology database. This field in the APPN0200 format is valid only when the node type enumeration is not 0.

Node flow reduction sequence number. The flow reduction sequence number (FRSN) for the node. This field in the APPN0200 format is valid only when the local node is a network node and the node type enumeration is 1 or 3 (the node is a network node or virtual node).

Node and transmission group FRSN numbers are not broadcast on topology data updates (TDUs) and not duplicated on every network node in the network. Each network node keeps its own node and transmission group FRSN for each node and transmission group in the topology database. FRSNs are used to minimize TDU exchanges with adjacent network nodes.

Node network ID. The network ID for the node.

Node network-qualified control point name. The network-qualified control point name for the node, in the format NETID.CPNAME.

Node type. The APPN node type of the node. This field in the APPN0100 format contains blanks when the node data valid indicator field is N. The valid values are:

*EN  APPN end node
*NN  APPN network node
*VN  APPN virtual node
There is no distinction made between APPN end nodes and low-entry networking (LEN) nodes. Any node defined as a LEN node is reported as *EN.

**Node type enumeration.** A number representing the APPN node type of the node. This field is used only for format APPN0200. The valid values are:

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Node Data not valid (this may occur when an entry only contains transmission groups owned by this node).</td>
</tr>
<tr>
<td>1</td>
<td>APPN network node</td>
</tr>
<tr>
<td>2</td>
<td>APPN end node</td>
</tr>
<tr>
<td>3</td>
<td>APPN virtual node</td>
</tr>
</tbody>
</table>

**Number of associated transmission group entries.** The number of transmission group (TG) entries reported for a node. When this value is not zero, the specified number of associated transmission groups are reported for the node. The first associated transmission group entry is reported on the same entry as the initial node entry, and any additional transmission groups are reported on subsequent entries.

**Number of end nodes reported.** The total number of end nodes returned in the list (end node entries containing valid node data).

**Number of nodes deleted since local node initialized.** The number of node entries deleted from the topology database since the local node was initialized.

**Number of nodes reported.** The total number of node entries returned in the list (entries containing valid node data).

**Number of transmission groups deleted since local node initialized.** The number of transmission group entries deleted from the topology database since the local node was initialized.

**Number of transmission groups reported for all end nodes.** The total number of associated transmission groups returned in the list owned by end nodes.

**Number of transmission groups reported for all network and virtual nodes.** The total number of associated transmission groups returned in the list owned by network nodes and virtual nodes.

**Reserved.** An ignored field.

**Transmission group characteristics.** The transmission group (TG) characteristics. See “Format of Transmission Group Characteristics” on page 15 for the structure. This field is valid only when the number of associated transmission group entries field is not zero.

**Transmission group destination control point name.** The control point name for the transmission group (TG) destination node. This field is blank when the number of associated transmission group entries field is zero.

**Transmission group destination network ID.** The network ID for transmission group (TG) destination node. This field is blank when the number of associated transmission group entries field is zero.

**Transmission group destination network-qualified control point name.** The network-qualified control point name for the transmission group destination node, in the format NETID.CPNAME. This field is blank when the number of associated transmission group entries field is zero.

**Transmission group entry time left.** The number of days left before the transmission group entry is deleted from the topology database. This field in the APPN0200 format is valid only when the number of associated transmission group entries field is not zero.
**Transmission group flags.** The format of the transmission group flags data is described in “Format of Transmission Group Flags Field” on page 14. This field is valid only when the number of associated transmission group entries field is not zero.

**Transmission group flow reduction sequence number.** The flow reduction sequence number (FRSN) for the transmission group. This field in the APPN0200 format is valid only when the number of associated transmission group entries field is not zero, the local node is a network node, and the transmission group reported is owned by a network node or virtual node.

Node and transmission group FRSN numbers are not broadcast on topology data updates (TDUs) and not duplicated on every network node in the network. Each network node keeps its own node and transmission group FRSN for each node and transmission group in the topology database. FRSNs are used to minimize TDU exchanges with adjacent network nodes.

**Transmission group number.** The transmission group (TG) number. This field is valid only when the number of associated transmission group entries field is not zero.

**Timestamp.** The machine timestamp (time of day) when reported.

**Timestamp of list data.** The machine timestamp of the list data.

**Timestamp when local node initialized.** The machine timestamp when the local node was initialized. The local node is initialized at initial program load (IPL) time and when the APPN node type changes.

**User space library name specified.** The user space library name specified to the API.

**User space library name used.** The actual user space library name used to report data.

**User space name specified.** The user space name specified to the API.

**User space name used.** The actual user space name used to report data.

**Format of Node Attributes Field**

The format of the node attributes field is described below.

<table>
<thead>
<tr>
<th>Offset</th>
<th>Bit</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Dec</td>
<td>Hex</td>
<td></td>
<td>Description</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>CHAR(4)</td>
<td>Resource sequence number</td>
</tr>
<tr>
<td>4</td>
<td>4</td>
<td>CHAR(1)</td>
<td>Route addition resistance</td>
</tr>
<tr>
<td>5</td>
<td>5</td>
<td>CHAR(1)</td>
<td>Node status</td>
</tr>
<tr>
<td>5</td>
<td>5</td>
<td>0</td>
<td>BIT(1)</td>
</tr>
<tr>
<td>5</td>
<td>5</td>
<td>1</td>
<td>BIT(1)</td>
</tr>
<tr>
<td>5</td>
<td>5</td>
<td>2</td>
<td>BIT(1)</td>
</tr>
<tr>
<td>5</td>
<td>5</td>
<td>3</td>
<td>BIT(2)</td>
</tr>
<tr>
<td>5</td>
<td>5</td>
<td>5</td>
<td>BIT(1)</td>
</tr>
<tr>
<td>5</td>
<td>5</td>
<td>6</td>
<td>BIT(2)</td>
</tr>
<tr>
<td>6</td>
<td>6</td>
<td>CHAR(1)</td>
<td>Node type and support</td>
</tr>
<tr>
<td>6</td>
<td>6</td>
<td>0</td>
<td>CHAR(1)</td>
</tr>
<tr>
<td>6</td>
<td>6</td>
<td>1</td>
<td>BIT(1)</td>
</tr>
<tr>
<td>6</td>
<td>6</td>
<td>2</td>
<td>BIT(1)</td>
</tr>
<tr>
<td>Offset</td>
<td>Dec</td>
<td>Hex</td>
<td>Bit</td>
</tr>
<tr>
<td>--------</td>
<td>-----</td>
<td>-----</td>
<td>-----</td>
</tr>
<tr>
<td>6</td>
<td>6</td>
<td>6</td>
<td>3</td>
</tr>
<tr>
<td>6</td>
<td>6</td>
<td>6</td>
<td>4</td>
</tr>
<tr>
<td>6</td>
<td>6</td>
<td>6</td>
<td>6</td>
</tr>
<tr>
<td>7</td>
<td>7</td>
<td>7</td>
<td></td>
</tr>
<tr>
<td>7</td>
<td>7</td>
<td>7</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>7</td>
<td>7</td>
<td>7</td>
<td>1</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>7</td>
<td>7</td>
<td>7</td>
<td>2</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>7</td>
<td>7</td>
<td>7</td>
<td>3</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>7</td>
<td>7</td>
<td>7</td>
<td>5</td>
</tr>
</tbody>
</table>

**Format of Transmission Group Flags Field**
The format of the transmission group flags data is described below.

<table>
<thead>
<tr>
<th>Offset</th>
<th>Dec</th>
<th>Hex</th>
<th>Bit</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td></td>
<td>CHAR(1)</td>
<td>Transmission group flags</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>BIT(1)</td>
<td>Link connection network indicator:</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0: The transmission group-Partner Node’s Network-Qualified CP/PU Name field does not identify a link connection network (such as a local area network)</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>1: The transmission group-Partner Node’s Network-Qualified CP/PU Name field identifies a link connection network</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>BIT(1)</td>
<td>Reserved</td>
</tr>
<tr>
<td>Offset</td>
<td>Dec</td>
<td>Hex</td>
<td>Bit</td>
<td>Type</td>
<td>Description</td>
</tr>
<tr>
<td>--------</td>
<td>-----</td>
<td>-----</td>
<td>-----</td>
<td>------</td>
<td>-------------</td>
</tr>
</tbody>
</table>
| 0      | 0   | 0   | 2   | BIT(1) | This transmission group supports high performance routing:  
0  No  
1  Yes |
| 0      | 0   | 0   | 3   | BIT(2) | Transmission group type:  
00  Boundary function based transmission group or APPN transmission group  
01  Interchange transmission group  
10  Virtual route based transmission group  
11  Reserved |
| 0      | 0   | 0   | 5   | BIT(1) | Intersubnetwork link indicator:  
0  This link is not an intersubnetwork link.  
1  This link is an intersubnetwork link. It defines a border between subnetworks. |
| 0      | 0   | 0   | 6   | BIT(1) | Reserved |
| 0      | 0   | 0   | 7   | BIT(1) | This transmission group goes to a node that supports the high performance routing transport tower:  
0  No  
1  Yes |

**Format of Transmission Group Characteristics**  
The format of the transmission group (TG) characteristics data is described below.

<table>
<thead>
<tr>
<th>Offset</th>
<th>Dec</th>
<th>Hex</th>
<th>Bit</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td></td>
<td>CHAR(4)</td>
<td>Resource sequence number (reserved except in topology database updates (TDUs)): a 32-bit binary value that uniquely identifies a topology update.</td>
</tr>
<tr>
<td>4</td>
<td>4</td>
<td>4</td>
<td></td>
<td>CHAR(1)</td>
<td>Status</td>
</tr>
</tbody>
</table>
| 4      | 4   | 4   | 0   | BIT(1) | Operational status:  
0  The transmission group is not operational  
1  The transmission group is operational |
| 4      | 4   | 4   | 1   | BIT(1) | Reserved |
| 4      | 4   | 4   | 2   | BIT(1) | Quiescing:  
0  The transmission group is not quiescing  
1  The transmission group is quiescing |
### Offset Table

<table>
<thead>
<tr>
<th>Offset</th>
<th>Dec</th>
<th>Hex</th>
<th>Bit</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>4</td>
<td>4</td>
<td>4</td>
<td>3</td>
<td>BIT(1)</td>
<td>CP-CP session support status:</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0      CP-CP sessions are supported on this transmission group.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>1      CP-CP sessions are not supported on this transmission group.</td>
</tr>
<tr>
<td>4</td>
<td>4</td>
<td>4</td>
<td>4</td>
<td>BIT(4)</td>
<td>Reserved</td>
</tr>
<tr>
<td>5</td>
<td>5</td>
<td>5</td>
<td></td>
<td>CHAR(1)</td>
<td>Effective capacity</td>
</tr>
<tr>
<td>6</td>
<td>6</td>
<td>6</td>
<td></td>
<td>CHAR(5)</td>
<td>Reserved</td>
</tr>
<tr>
<td>11</td>
<td>B</td>
<td></td>
<td></td>
<td>CHAR(1)</td>
<td>Cost per connect time</td>
</tr>
<tr>
<td>12</td>
<td>C</td>
<td></td>
<td></td>
<td>CHAR(1)</td>
<td>Cost per byte transmitted</td>
</tr>
<tr>
<td>13</td>
<td>D</td>
<td></td>
<td></td>
<td>CHAR(1)</td>
<td>Reserved</td>
</tr>
<tr>
<td>14</td>
<td>E</td>
<td></td>
<td></td>
<td>CHAR(1)</td>
<td>Security</td>
</tr>
<tr>
<td>15</td>
<td>F</td>
<td></td>
<td></td>
<td>CHAR(1)</td>
<td>Propagation delay of the transmission group</td>
</tr>
<tr>
<td>16</td>
<td>10</td>
<td></td>
<td></td>
<td>CHAR(1)</td>
<td>Modem class</td>
</tr>
<tr>
<td>17</td>
<td>11</td>
<td></td>
<td></td>
<td>CHAR(1)</td>
<td>User-defined parameter 1</td>
</tr>
<tr>
<td>18</td>
<td>12</td>
<td></td>
<td></td>
<td>CHAR(1)</td>
<td>User-defined parameter 2</td>
</tr>
<tr>
<td>19</td>
<td>13</td>
<td></td>
<td></td>
<td>CHAR(1)</td>
<td>User-defined parameter 3</td>
</tr>
</tbody>
</table>

### Error Messages

<table>
<thead>
<tr>
<th>Message ID</th>
<th>Error Message Text</th>
</tr>
</thead>
<tbody>
<tr>
<td>CPF24B4 E</td>
<td>Severe error while addressing parameter list.</td>
</tr>
<tr>
<td>CPF3C21 E</td>
<td>Format name &amp;1 is not valid.</td>
</tr>
<tr>
<td>CPF3CAA E</td>
<td>List is too large for user space &amp;1.</td>
</tr>
<tr>
<td>CPF3C90 E</td>
<td>Literal value cannot be changed.</td>
</tr>
<tr>
<td>CPF3CF1 E</td>
<td>Error code parameter not valid.</td>
</tr>
<tr>
<td>CPF8100 E</td>
<td>All CPF81xx messages could be returned. xx is from 01 to FF.</td>
</tr>
<tr>
<td>CPF91C2 E</td>
<td>Queue type must be *DTAQ or *USRQ.</td>
</tr>
<tr>
<td>CPF91C3 E</td>
<td>Internal processing error.</td>
</tr>
<tr>
<td>CPF91C4 E</td>
<td>Queue &amp;5/&amp;4 with type &amp;6 already registered.</td>
</tr>
<tr>
<td>CPF91C5 E</td>
<td>Queue &amp;2/&amp;1 not valid.</td>
</tr>
<tr>
<td>CPF91C7 E</td>
<td>Options list not valid.</td>
</tr>
<tr>
<td>CPF91C8 E</td>
<td>Replace registration must be *YES or *NO.</td>
</tr>
<tr>
<td>CPF9800 E</td>
<td>All CPF98xx messages could be signaled. xx is from 01 to FF.</td>
</tr>
</tbody>
</table>

API introduced: V2R3

### SNA/Management Services Transport APIs

The SNA/Management services Transport APIs are:

- "Change Mode Name (QNMCHGMN) API" on page 21 (QNMCHGMN) sets the APPC mode name used when sending requests.
Systems Network Architecture (SNA) Management Services Transport (SNA/MS Transport) functions are used to support the sending and receiving of management services data between systems in an SNA network. The network can include iSeries™ systems, Operating System/2™ (OS/2) and NetView® licensed programs, and other platforms that support the SNA/Management Services architecture.

The SNA/Management Services functions provided on the iSeries system include:

- The transport of network management data in APPN networks
- The maintenance of node relationships for network management

The APIs allow a network management application running on one system to send data to and receive data from a network management application running on another system in an APPN network. The APIs are a callable interface that allow the application to be notified about asynchronous events, such as incoming data, by way of a notification placed on a data queue.

Some examples of IBM® applications that use SNA/Management Services Transport APIs are:

- Alerts
- Problem reporting
- Remote problem analysis
- Program temporary fix (PTF) ordering

In large networks, the number of sessions needed to support the various network management applications could become burdensome without session concentration. SNA/Management Services Transport APIs reduce the number of SNA LU 6.2 sessions that would normally be used to transmit data. This support multiplexes or transmits all of the network management data from all the applications in a network node domain (network node and attached end nodes) on a single session to applications in another domain.

This means that data transmitted from an end node is always sent to its network node server first. Then, the SNA/Management Services Transport support on the network node server routes the data to its proper destination.
Using the SNA/Management Services Transport APIs

SNA/Management Services is used by two types of applications: a source application and a target application. A source application sends requests to and receives replies from a target application. Similarly, a target application receives requests from and sends replies to a source application.

For a target application to receive requests from a source application, the target application must perform the following operations:

1. Create a data queue, using the Create Data Queue (CRTDTAQ) command, to allow SNA/Management Services Transport support to indicate incoming data (the MAXLEN parameter must be 80 or greater and the SEQ parameter must be *FIFO).
2. Start an application specifying the data queue just created, using the Start Application (QNMSTRAP) API. The application can then receive a handle, which will be used by the application on all following API calls. The handle is an identifier that represents the application being worked on and is unique within a job.
3. Register an application, using the Register Application (QNMREGAP) API. This notifies SNA/Management Services Transport support that the application is able to receive requests from source applications.
4. Wait for a request to arrive using the Receive Data Queue (QRCVDTAQ) API. See Receive Data Queue (QRCVDTAQ) API for more information.

When the request arrives, the target application receives an entry from the data queue using the QRCVDTAQ API. The target application then uses the request identifier in the entry to receive the data (call the Receive Data (QNMRCVDT) API). The data queue entry has the following format.

### Entry Format

<table>
<thead>
<tr>
<th>Offset</th>
<th>Value</th>
<th>Field</th>
</tr>
</thead>
<tbody>
<tr>
<td>Dec</td>
<td>Hex</td>
<td>Type</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>CHAR(10)</td>
</tr>
<tr>
<td>10</td>
<td>A</td>
<td>CHAR(2)</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>12</td>
<td>C</td>
<td>BINARY(4)</td>
</tr>
<tr>
<td>16</td>
<td>10</td>
<td>CHAR(53)</td>
</tr>
<tr>
<td>69</td>
<td>45</td>
<td>CHAR(11)</td>
</tr>
</tbody>
</table>

### Field Descriptions

This topic describes the fields returned in further detail. The fields are listed in alphabetical order.

**Entry type.** This entry was created by SNA/Management Services Transport.

**Handle.** This value specifies the handle of the application associated with the data queue.

**Incoming data.** The incoming data has arrived and that the Receive Data (QNMRCVDT) API should be called.

**Request identifier.** Identifies incoming or transmitted data. This field is used when calling Receive Data (QNMRCVDT) and Receive Operation Completion (QNMRCVOC).

**Reserved.** An ignored field.
**Send complete.** A previous send operation has completed and that the Receive Operation Completion (QNMRCVOC) API should be called.

Then, depending on whether or not the request requires a reply, the target application may need to send a reply to the source application. A reply is sent using the Send Reply (QNMSNDRP) API.

A single request may require more than one reply. If a request is not recognized, a single error message may be sent instead of a reply using the Send Error (QNMSNDER) API.

Also, the source application must start the application (call the QNMSTRAP API), but the source does not need to create a data queue or register itself with SNA/Management Services Transport support. The source application can send a request to the target application, using the Send Request (QNMSNDRQ) API. The QNMSNDRQ API returns a request identifier that is used to receive any associated replies.

The source application receives any expected replies (call the QNMRCVDT API) with the request identifier parameter specified as the request identifier returned when the Send Request (QNMSNDRQ) API was called.

Both the target and the source applications use the End Application (QNMENDAP) API to end the management services transport support. The target application may optionally use the Deregister Application (QNMDRGAP) API before ending. However, the QNMENDAP API automatically performs a deregister operation.

The following example shows how these SNA/Management Services Transport APIs work together.

**Applications Using SNA/Management Services Transport APIs**
Data

The types of data handled by SNA/Management Services may be:
1. A request with a single reply expected (for example, a request for current information).
2. A request without a reply (for example, an alert)
3. A request with multiple replies expected
4. An unrecognized request returning an error message

The list correlates to the numbers on the left in the following example, which shows the flow of data requests and replies depending on the parameters specified.

Data Types Handled by SNA/Management Services Transport APIs

Notes:
1. Arrival of requests is not guaranteed unless a reply is requested. The reply acts as a form of acknowledgment.
2. The maximum amount of application data in a single request or reply is 31,739 bytes.

Routing
SNA/Management Services Transport performs all routing based on the network ID, control point name (or a logical unit name may be used), and application name. Applications must be registered with SNA/Management Services to receive unsolicited requests.

Communications from a network node to an end node is normally performed through the end node’s network node server. In the network node server, SNA/Management Services Transport provides intermediate routing functions between the end node and the network node. This is separate from the intermediate routing provided by APPN network node services.

By default, data sent using the SNA/Management Services Transport APIs uses sessions associated with system mode names CPSVCMG and SNASVCMG. CPSVCMG sessions are used between an end node and its network node server. SNASVCMG sessions are used between network nodes.

When data is sent from an end node to a network node that is not its network node server, its network node server performs intermediate routing between the CPSVCMG session and the SNASVCMG session.

Other sessions can be established by specifying a different mode name with the Change Mode Name (QNMCHGMN) API. When you change the mode name, APPN network node services performs the intermediate routing function rather than SNA/Management Services Transport. This is known as direct routing to SNA/Management Services Transport.

Change Mode Name (QNMCHGMN) API

Required Parameter Group:

<table>
<thead>
<tr>
<th></th>
<th>Parameter</th>
<th>Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Handle</td>
<td>Input</td>
</tr>
<tr>
<td>2</td>
<td>Mode name</td>
<td>Input</td>
</tr>
<tr>
<td>3</td>
<td>Error code</td>
<td>I/O</td>
</tr>
</tbody>
</table>

Default Public Authority: *USE

Threadsafe: No

The Change Mode Name (QNMCHGMN) API, an SNA/Management Services Transport API, sets the APPC mode name used when sending requests.

When a specific mode name is specified, direct routing is used. The result is a direct session from the source system to the target system.

Data sent using the system mode name SNASVCMG is sent at network priority, which means that data specifying this mode flows before any other data. Setting the mode to something other than SNASVCMG can lessen the effect on network performance and allow bulk data transfer. If large amounts of data are to be sent relative to the speed of the communications medium, then consider another mode. A mode object specified by its name in the mode name parameter must exist at the time this API is called.

Authorities and Locks

None.
**Required Parameter Group**

**Handle**

INPUT; BINARY(4)

The unique identifier for this application, which was returned by the Start Application (QNMSSTRAP) API.

**Mode name**

INPUT; CHAR(8)

The APPC mode name used on subsequent requests. Special values are:

*NETATR The current default mode name specified in the network attributes.
*DFTRTG A CPSVCMG session is used between end nodes and network nodes, and an SNASVCMG session is used between network nodes. CPSVCMG may not be specified. *DFTRTG is the default value, and it may require multiple APPN sessions to transport the data. In this case, the network node server performs SNA/Management Services routing for all end node traffic. This results in fewer sessions in the network and a slower response time.

**Note:** Some non-iSeries products only support *DFTRTG.

**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**

<table>
<thead>
<tr>
<th>Message ID</th>
<th>Error Message Text</th>
</tr>
</thead>
<tbody>
<tr>
<td>CPF24B4 E</td>
<td>Severe error while addressing parameter list.</td>
</tr>
<tr>
<td>CPF3C90 E</td>
<td>Literal value cannot be changed.</td>
</tr>
<tr>
<td>CPF3CF1 E</td>
<td>Error code parameter not valid.</td>
</tr>
<tr>
<td>CPF7AE2 E</td>
<td>Handle &amp;1 not found.</td>
</tr>
<tr>
<td>CPF7AE4 E</td>
<td>Mode name &amp;3 not valid.</td>
</tr>
<tr>
<td>CPF9872 E</td>
<td>Program or service program &amp;1 in library &amp;2 ended. Reason code &amp;3.</td>
</tr>
</tbody>
</table>

**API introduced:** V2R2

**Deregister Application (QNMDRGAP) API**

Required Parameter Group:

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

Default Public Authority: *USE

Threading: No

The Deregister Application (QNMDRGAP) API, an SNA/Management Services Transport API, removes the registration of the application name associated with a handle. The QNMDRGAP API only deregisters the application; the application can continue to send requests as long as replies are expected.

Only registered applications can receive requests from remote systems. After an application is deregistered, the remote applications are sent error messages notifying them that the application is not
available to receive requests. In addition, it can no longer receive error messages from requests that did not request a reply. Therefore, the application should not send any request-only data.

The QNMDRGAP API is used by applications that do not want to continue receiving requests. The End Application (QNMENDAP) API automatically removes registration before ending the application instance. See the "End Application (QNMENDAP) API" for more information.

**Authorities and Locks**
None.

**Required Parameter Group**

**Handle**

INPUT; BINARY(4)

The unique identifier for this application, which was returned by the Start Application (QNMSTRAP) API.

**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**
---|---
CPF24B4 E | Severe error while addressing parameter list. | 
CPF3C90 E | Literal value cannot be changed. | 
CPF3CF1 E | Error code parameter not valid. | 
CPF7ADC E | Internal processing error. | 
CPF7AEE E | Application &2 not registered. | 
CPF7AE2 E | Handle &1 not found. | 
CPF9872 E | Program or service program &1 in library &2 ended. Reason code &3. | 

API introduced: V2R2

---

**End Application (QNMENDAP) API**

**Required Parameter Group:**

<p>| | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Handle</td>
<td>Input</td>
</tr>
<tr>
<td>2</td>
<td>Error code</td>
<td>I/O</td>
</tr>
</tbody>
</table>

Default Public Authority: *USE

Threadsafe: No

The End Application (QNMENDAP) API, an SNA/Management Services Transport API, ends support for the application associated with the handle. If the local application name is currently registered, its registration is automatically removed.

Applications must ensure that this API is performed when a handle is no longer needed. If you do not use QNMENDAP, the application automatically ends when the job is complete.
Authorities and Locks
None.

Required Parameter Group

Handle
INPUT; BINARY(4)
The unique identifier for this application, which was returned by the Start Application (QNMSTRAP) API.

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

<table>
<thead>
<tr>
<th>Message ID</th>
<th>Error Message Text</th>
</tr>
</thead>
<tbody>
<tr>
<td>CPF24B4 E</td>
<td>Severe error while addressing parameter list.</td>
</tr>
<tr>
<td>CPF3C90 E</td>
<td>Literal value cannot be changed.</td>
</tr>
<tr>
<td>CPF3CF1 E</td>
<td>Error code parameter not valid.</td>
</tr>
<tr>
<td>CPF7AE2 E</td>
<td>Handle &amp;1 not found.</td>
</tr>
<tr>
<td>CPF9872 E</td>
<td>Program or service program &amp;1 in library &amp;2 ended. Reason code &amp;3.</td>
</tr>
</tbody>
</table>

API introduced: V2R1

Receive Data (QNMRCVDT) API

Required Parameter Group:

1. Handle
2. Receiver variable
3. Length of receiver variable
4. Request identifier
5. Remote application name
6. Data type
7. Wait time
8. Error code

Input
Input
Input
Input
Output
Output
Input
I/O

Binary(4)
Binary(*)
Binary(4)
CHAR(53)
CHAR(24)
CHAR(10)
Binary(4)
CHAR(*)

Default Public Authority: *USE
Threadsafe: No

The Receive Data (QNMRCVDT) API, an SNA/Management Services Transport API, receives a particular request, reply, or error message.

The receiver variable data area is not changed unless it is large enough to hold all of the data received. If the receiver variable is not large enough to hold the received data, then an error message is returned indicating how much more space is needed. The application can allocate a new buffer size greater than or equal to the bytes available and call the Receive Data (QNMRCVDT) API again. If the received data is smaller than the receiver variable, any data in the unused portion of the variable is unchanged.

Authorities and Locks
None.
**Required Parameter Group**

**Handle**

INPUT; BINARY(4)

The unique identifier for this application, which was returned by the Start Application (QNMSTRAP) API.

**Receiver variable**

OUTPUT; CHAR(*)

The variable in which this API returns the data. This structure includes the bytes returned and bytes available in addition to the received data.

The format of the receiver variable is:

<table>
<thead>
<tr>
<th>Offset</th>
<th>Type</th>
<th>Field</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>BINARY(4)</td>
<td>Bytes returned</td>
</tr>
<tr>
<td>4</td>
<td>BINARY(4)</td>
<td>Bytes available</td>
</tr>
<tr>
<td>8</td>
<td>CHAR(*)</td>
<td>Received data</td>
</tr>
</tbody>
</table>

The bytes returned field specifies the length of the bytes actually returned. The bytes available field specifies the total length of data available to be returned.

**Length of receiver variable**

INPUT; BINARY(4)

The size of the receiver variable parameter, which is the maximum amount of data that can be returned in the receiver variable.

**Request identifier**

INPUT; CHAR(53)

The request identifier of the data being received.

*PRV The last request identifier used (for example, the one returned on the Send Request (QNMSNDRQ) API).

**Remote application name**

OUTPUT; CHAR(24)

The name of the remote application that sent the data. The first 8 characters contain the network ID, the second 8 characters contain the control point name (or the logical unit name may be used), and the third 8 characters contain the application name of the remote application.

**Data type**

OUTPUT; CHAR(10)

The type of data returned.

*RQS A request was received. No reply is expected.

*RQSRPY A request was received. A reply is expected.

*RPYCPL A complete reply was received. This is either the last or only reply.

*RPYINCPL An incomplete reply was received. Additional Receive Data (QNMRCVDT) operations should be performed.

*NODATA No data was received. Either an error occurred or the wait time elapsed.

**Wait time**

INPUT; BINARY(4)
The amount of time the application waits for the data to be received.

-1  Waits for data to be received (no matter how long it takes) or for a condition such that the reply cannot be received (for example, a communications failure).

0-99999  The number of seconds the application waits.

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

<table>
<thead>
<tr>
<th>Message ID</th>
<th>Error Message Text</th>
</tr>
</thead>
<tbody>
<tr>
<td>CPF24B4</td>
<td>Severe error while addressing parameter list.</td>
</tr>
<tr>
<td>CPF3C24</td>
<td>Length of the receiver variable is not valid.</td>
</tr>
<tr>
<td>CPF3C90</td>
<td>Literal value cannot be changed.</td>
</tr>
<tr>
<td>CPF3CF1</td>
<td>Error code parameter not valid.</td>
</tr>
<tr>
<td>CPF7AC0</td>
<td>&amp;3.&amp;4 cannot receive data at this time.</td>
</tr>
<tr>
<td>CPF7AC4</td>
<td>Control point &amp;3.&amp;4 rejected data.</td>
</tr>
<tr>
<td>CPF7ADC</td>
<td>Internal processing error.</td>
</tr>
<tr>
<td>CPF7ADD</td>
<td>Operation did not complete.</td>
</tr>
<tr>
<td>CPF7ADF</td>
<td>Session failure. Cannot send data at this time.</td>
</tr>
<tr>
<td>CPF7AEA</td>
<td>Application on control point &amp;3.&amp;4 failed.</td>
</tr>
<tr>
<td>CPF7AEC</td>
<td>Wait time &amp;3 not between -1 and 99999.</td>
</tr>
<tr>
<td>CPF7AE0</td>
<td>Function requested not supported by remote system.</td>
</tr>
<tr>
<td>CPF7AE2</td>
<td>Handle &amp;1 not found.</td>
</tr>
<tr>
<td>CPF7AE3</td>
<td>Management Services transport operation not permitted.</td>
</tr>
<tr>
<td>CPF7AE6</td>
<td>Communication with control point &amp;3.&amp;4 failed.</td>
</tr>
<tr>
<td>CPF7AE7</td>
<td>Remote application program &amp;5 not found.</td>
</tr>
<tr>
<td>CPF7AE8</td>
<td>Receiver variable too small.</td>
</tr>
<tr>
<td>CPF7AE9</td>
<td>Request identifier not valid.</td>
</tr>
<tr>
<td>CPF9872</td>
<td>Program or service program &amp;1 in library &amp;2 ended. Reason code &amp;3.</td>
</tr>
</tbody>
</table>

API introduced: V2R2

Receive Operation Completion (QNMRCVOC) API

Required Parameter Group:

<p>| | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Handle</td>
<td>Input</td>
</tr>
<tr>
<td>2</td>
<td>Request identifier</td>
<td>Input</td>
</tr>
<tr>
<td>3</td>
<td>Remote application name</td>
<td>Output</td>
</tr>
<tr>
<td>4</td>
<td>Error code</td>
<td>I/O</td>
</tr>
</tbody>
</table>

Default Public Authority: USE

Threadsafe: No

The Receive Operation Completion (QNMRCVOC) API, an SNA/Management Services Transport API, receives an operation completion, which allows an application to determine if a send operation completed successfully. This is only used by applications that do not wait for a send operation to complete (wait time parameter equals 0). This API must be used after an application receives an entry on the data queue confirming the completion of an operation.
If the operation did not complete successfully, an error code is returned or an exception is signaled based on i5/OS error-handling rules.

**Authorities and Locks**
None.

**Required Parameter Group**

**Handle**

- **INPUT; BINARY(4)**
  
  The unique identifier for this application, which was returned by the Start Application (QNMSTRAP) API.

**Request identifier**

- **INPUT; CHAR(53)**
  
  The request identifier of the operation completion reply that is to be received.

  *PRV  
  
  The last request identifier used (for example, the one returned on the Send Request (QNMSNDRQ) API).

**Remote application name**

- **OUTPUT; CHAR(24)**
  
  The name of the remote application that received the data. The first 8 characters contain the network ID, the second 8 characters contain the control point name (or the logical unit name may be used), and the third 8 characters contain the application name of the remote application.

**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**

<table>
<thead>
<tr>
<th>Message ID</th>
<th>Error Message Text</th>
</tr>
</thead>
<tbody>
<tr>
<td>CPF24B4 E</td>
<td>Severe error while addressing parameter list.</td>
</tr>
<tr>
<td>CPF3C90 E</td>
<td>Literal value cannot be changed.</td>
</tr>
<tr>
<td>CPF3CF1 E</td>
<td>Error code parameter not valid.</td>
</tr>
<tr>
<td>CPF7AC0 E</td>
<td>&amp;3.&amp;4 cannot receive data at this time.</td>
</tr>
<tr>
<td>CPF7AC2 E</td>
<td>Previous send operation not complete.</td>
</tr>
<tr>
<td>CPF7ADC E</td>
<td>Internal processing error.</td>
</tr>
<tr>
<td>CPF7ADF E</td>
<td>Session failure. Cannot send data at this time.</td>
</tr>
<tr>
<td>CPF7AE0 E</td>
<td>Function requested not supported by remote system.</td>
</tr>
<tr>
<td>CPF7AE2 E</td>
<td>Handle &amp;1 not found.</td>
</tr>
<tr>
<td>CPF7AE5 E</td>
<td>Control point &amp;3.&amp;4 not found.</td>
</tr>
<tr>
<td>CPF7AE6 E</td>
<td>Communication with control point &amp;3.&amp;4 failed.</td>
</tr>
<tr>
<td>CPF7AE7 E</td>
<td>Remote application program &amp;5 not found.</td>
</tr>
<tr>
<td>CPF7AE9 E</td>
<td>Request identifier not valid.</td>
</tr>
<tr>
<td>CPF9872 E</td>
<td>Program or service program &amp;1 in library &amp;2 ended. Reason code &amp;3.</td>
</tr>
</tbody>
</table>

API introduced: V2R2
Register Application (QNMREGAP) API

Required Parameter Group:

1. Handle  
   Input  
   Binary(4)

2. Category  
   Input  
   Char(8)

3. Application type  
   Input  
   Char(10)

4. Replace Registration  
   Input  
   Char(10)

5. Error code  
   I/O  
   Char(*)

Default Public Authority: *USE
Threadsafe: No

The Register Application (QNMREGAP) API, an SNA/Management Services Transport API, registers the application name specified in the previous start application operation so that it may receive unsolicited requests. A given application name may only be registered once on the system.

Applications must be registered to receive requests and error messages when a reply is not expected. The application name used is specified on the Start Application (QNMSTRAP) API. See the "Start Application (QNMSTRAP) API" on page 35 for more information.

The only applications that do not need to register are those issuing requests and expecting replies.

If the value of the replace registration parameter is *YES, a previously registered application with the same name can no longer receive requests.

Authorities and Locks
None.

Required Parameter Group

Handle
   INPUT; BINARY(4)
   The unique identifier for this application, which was returned by the Start Application (QNMSTRAP) API.

Category
   INPUT; CHAR(8)
   The SNA/Management Services function set group with which the application is associated. The only allowed value is:
   *NONE  Not associated with any category.

Application type
   INPUT; CHAR(10)
   The role the application is performing:
   *EPAPP  An entry point application.
   *FPAPP  A focal point application.

Replace registration
   INPUT; CHAR(10)
   Whether or not this registration should replace a previous registration that has the same local application name.
This registration should replace any previous registration with the same local application name. Any other SNA/Management Services program that previously registered with the same local application name can no longer receive incoming requests.

Do not replace previous registrations. If another SNA/Management Services program in the same or different job is already registered for this local application name, then it continues to receive incoming requests. Error message CPF7AED is issued if that application is already registered.

**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**

<table>
<thead>
<tr>
<th>Message ID</th>
<th>Error Message Text</th>
</tr>
</thead>
<tbody>
<tr>
<td>CPF24B4 E</td>
<td>Severe error while addressing parameter list.</td>
</tr>
<tr>
<td>CPF3C90 E</td>
<td>Literal value cannot be changed.</td>
</tr>
<tr>
<td>CPF3CF1 E</td>
<td>Error code parameter not valid.</td>
</tr>
<tr>
<td>CPF7ADB E</td>
<td>Replace registration value &amp;3 not valid.</td>
</tr>
<tr>
<td>CPF7ADC E</td>
<td>Internal processing error.</td>
</tr>
<tr>
<td>CPF7AEB E</td>
<td>Category value &amp;3 not valid.</td>
</tr>
<tr>
<td>CPF7AED E</td>
<td>Application &amp;2 already registered.</td>
</tr>
<tr>
<td>CPF7AEF E</td>
<td>Application type &amp;3 not valid.</td>
</tr>
<tr>
<td>CPF7AE2 E</td>
<td>Handle &amp;1 not found.</td>
</tr>
<tr>
<td>CPF9872 E</td>
<td>Program or service program &amp;1 in library &amp;2 ended. Reason code &amp;3.</td>
</tr>
</tbody>
</table>

API introduced: V2R2

The Retrieve Mode Name (QNMRTVMN) API, an SNA/Management Services Transport API, retrieves the APPC mode name currently being used for the purpose of sending requests. The mode name retrieved is the one currently associated with the application identified by the handle.

**Authorities and Locks**

None.

**Required Parameter Group**

**Handle**

INPUT; BINARY(4)

The unique identifier for this application, which was returned by the Start Application (QNMSTRAP) API.
Mode name

OUTPUT; CHAR(8)

The APPC mode name that is used when a request is sent. The special value is:

*DFTRTG A CPSVCMG session is used between end nodes and network nodes, and a SNASVCMG session is used between network nodes. *DFTRTG is the default parameter, and it may require multiple APPN sessions to transport the data. In this case, the network node server performs SNA/Management Services routing for all end node traffic. This results in fewer total sessions in the network and a slower response time.

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

<table>
<thead>
<tr>
<th>Message ID</th>
<th>Error Message Text</th>
</tr>
</thead>
<tbody>
<tr>
<td>CPF24B4 E</td>
<td>Severe error while addressing parameter list.</td>
</tr>
<tr>
<td>CPF3C90 E</td>
<td>Literal value cannot be changed.</td>
</tr>
<tr>
<td>CPF3CF1 E</td>
<td>Error code parameter not valid.</td>
</tr>
<tr>
<td>CPF7AE2 E</td>
<td>Handle &amp;1 not found.</td>
</tr>
<tr>
<td>CPF9872 E</td>
<td>Program or service program &amp;1 in library &amp;2 ended. Reason code &amp;3.</td>
</tr>
</tbody>
</table>

API introduced: V2R2

Send Error (QNMSNDER) API

Required Parameter Group:

<p>| | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Handle</td>
<td>Input</td>
</tr>
<tr>
<td>2</td>
<td>Request identifier</td>
<td>Input</td>
</tr>
<tr>
<td>3</td>
<td>Application error code</td>
<td>Input</td>
</tr>
<tr>
<td>4</td>
<td>Error code</td>
<td>I/O</td>
</tr>
</tbody>
</table>

Default Public Authority: *USE

Threadsafe: No

The Send Error (QNMSNDER) API, an SNA/Management Services Transport API, sends an SNA/Management Services error message to the remote application. This is used by the target application if the remote system is unable to process a request because the request data is not recognized.

If the application receiving the error message is on an iSeries server, it will receive the error message using the Receive Data (QNMRCVDT) API. This operation results in a CPF7AC4 error message being sent, and no data is returned.

Authorities and Locks

None.

Required Parameter Group

Handle

INPUT; BINARY(4)
The unique identifier for this application, which was returned by the Start Application (QNMSTRAP) API.

Request identifier
  INPUT; CHAR(53)

  The identifier for the request that contained an error.

*PRV  The last request identifier used (for example, the one specified on the Receive Data (QNMRCVDT) API).

Application error code
  INPUT; CHAR(10)

  The type of failure specified:

*BADDATA  The application is unable to interpret the data received.
*CANCEL  The application has canceled the processing of the this request.

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

<table>
<thead>
<tr>
<th>Message ID</th>
<th>Error Message Text</th>
</tr>
</thead>
<tbody>
<tr>
<td>CPF24B4 E</td>
<td>Severe error while addressing parameter list.</td>
</tr>
<tr>
<td>CPF3C90 E</td>
<td>Literal value cannot be changed.</td>
</tr>
<tr>
<td>CPF3CF1 E</td>
<td>Error code parameter not valid.</td>
</tr>
<tr>
<td>CPF7AC0 E</td>
<td>&amp;3.&amp;4 cannot receive data at this time.</td>
</tr>
<tr>
<td>CPF7ADC E</td>
<td>Internal processing error.</td>
</tr>
<tr>
<td>CPF7ADD E</td>
<td>Operation did not complete.</td>
</tr>
<tr>
<td>CPF7ADF E</td>
<td>Session failure. Cannot send data at this time.</td>
</tr>
<tr>
<td>CPF7AD1 E</td>
<td>Application error code value &amp;3 not valid.</td>
</tr>
<tr>
<td>CPF7AE2 E</td>
<td>Handle &amp;1 not found.</td>
</tr>
<tr>
<td>CPF7AE6 E</td>
<td>Communication with control point &amp;3.&amp;4 failed.</td>
</tr>
<tr>
<td>CPF7AE9 E</td>
<td>Request identifier not valid.</td>
</tr>
<tr>
<td>CPF9872 E</td>
<td>Program or service program &amp;1 in library &amp;2 ended. Reason code &amp;3.</td>
</tr>
</tbody>
</table>

API introduced: V2R2
The Send Reply (QNMSNDRP) API, an SNA/Management Services Transport API, sends a reply to a request that was received from a source application. Single or multiple replies may be sent for a particular request. The received request must have indicated that a reply is expected.

Multiple replies are sent by calling the Send Reply (QNMSNDRP) API one or more times. To send multiple replies, the reply type parameter is set to incomplete until the last reply is sent, when it is set to complete.

If the wait time is 0, then an entry is placed on the data queue when the operation is complete. If multiple replies are sent without waiting (wait time equals 0), multiple entries are placed on the data queue.

The same mode name used with the Send Request (QNMSNDRQ) API is used for the reply. See the “Send Request (QNMSNDRQ) API” on page 33 for more information about sending a request.

**Required Parameter Group**

**Handle**
- INPUT; BINARY(4)
  - The unique identifier for this application.

**Request identifier**
- INPUT; CHAR(53)
  - The request identifier of the corresponding request. The request identifier is returned when the original request is received.

*PRV  The last request identifier used (for example, the one specified on the Receive Data (QNMRCVDT) API).

**Send buffer**
- INPUT; CHAR(*)
  - The data being sent.

**Length of send buffer**
- INPUT; BINARY(4)
  - The size of the data being sent. The send buffer can range in size from 0 through 31739.

**Reply type**
- INPUT; CHAR(10)
  - The type of data being sent:
  
  *RPYCPL  The last or only reply (the reply is complete).
  *RPYINCPL Additional replies will be sent (the reply is incomplete).

**Wait time**
- INPUT; BINARY(4)
  - The amount of time the application waits for the send operation to complete.

  -1  Waits for the send operation to complete or for a condition such that the operation cannot complete (for example, a communications failure).
  0  The application does not wait for the operation to complete.
  1-99999  The number of seconds the application waits.

**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

<table>
<thead>
<tr>
<th>Message ID</th>
<th>Error Message Text</th>
</tr>
</thead>
<tbody>
<tr>
<td>CPF24B4 E</td>
<td>Severe error while addressing parameter list.</td>
</tr>
<tr>
<td>CPF3C90 E</td>
<td>Literal value cannot be changed.</td>
</tr>
<tr>
<td>CPF3CF1 E</td>
<td>Error code parameter not valid.</td>
</tr>
<tr>
<td>CPF7AC0 E</td>
<td>&amp;3.&amp;4 cannot receive data at this time.</td>
</tr>
<tr>
<td>CPF7AC1 E</td>
<td>Reply cannot be sent.</td>
</tr>
<tr>
<td>CPF7ADC E</td>
<td>Internal processing error.</td>
</tr>
<tr>
<td>CPF7ADD E</td>
<td>Operation did not complete.</td>
</tr>
<tr>
<td>CPF7ADF E</td>
<td>Session failure. Cannot send data at this time.</td>
</tr>
<tr>
<td>CPF7AD5 E</td>
<td>Reply type value &amp;3 not valid.</td>
</tr>
<tr>
<td>CPF7AEC E</td>
<td>Wait time &amp;3 not between -1 and 99999.</td>
</tr>
<tr>
<td>CPF7AE2 E</td>
<td>Handle &amp;1 not found.</td>
</tr>
<tr>
<td>CPF7AE6 E</td>
<td>Communication with control point &amp;3.&amp;4 failed.</td>
</tr>
<tr>
<td>CPF7AE9 E</td>
<td>Request identifier not valid.</td>
</tr>
<tr>
<td>CPF9872 E</td>
<td>Program or service program &amp;1 in library &amp;2 ended. Reason code &amp;3.</td>
</tr>
</tbody>
</table>

API introduced: V2r1.1

Send Request (QNMSNDRQ) API

Required Parameter Group:

|   | Handle          |   | Remote application name |   | Request Identifier |   | Send buffer |   | Length of send buffer |   | Request type |   | Post reply |   | Wait time |   | Error code |
|---|-----------------|---|-------------------------|---|--------------------|---|--------------|---|------------------------|---|--------------|---|-----------|---|-----------|
| 1 | Handle          |   | Input                   |   | Binary(4)         |   | Input        |   | Char(24)               |   |             |   |           |   |           |
| 2 | Remote application name |   | Input                   |   | Char(24)         |   | Output      |   | Char(53)               |   |             |   |           |   |           |
| 3 | Request Identifier |   | Output                  |   | Char(53)         |   | Input        |   | Char(*)                |   |             |   |           |   |           |
| 4 | Send buffer     |   | Input                   |   | Char(53)         |   | Input        |   | Char(*)                |   |             |   |           |   |           |
| 5 | Length of send buffer |   | Input                   |   | Char(*)         |   | Input        |   | Char(10)               |   |             |   |           |   |           |
| 6 | Request type    |   | Input                   |   | Char(10)         |   | Input        |   | Char(10)               |   |             |   |           |   |           |
| 7 | Post reply      |   | Input                   |   | Char(10)         |   | Input        |   | Char(*)                |   |             |   |           |   |           |
| 8 | Wait time       |   | Input                   |   | Char(10)         |   | Input        |   | Char(*)                |   |             |   |           |   |           |
| 9 | Error code      |   | I/O                     |   | Char(*)         |   |             |   |                       |   |             |   |           |   |           |

Default Public Authority: *USE
Threadsafe: No

The Send Request (QNMSNDRQ) API, an SNA/Management Services Transport API, sends a request to a remote application. The source application program using this API can indicate if a reply should be returned. If request-only data is sent, the source application will not know if the request data was successfully delivered. To confirm delivery, the source application must request a reply.

When the Send Request (QNMSNDRQ) API operation completes, the remote application may or may not be present. When the send request is complete, the request has been sent from the local system. If a reply is expected but the remote application does not exist, an error code is returned on the subsequent receive operation.

If a reply is not expected, the post reply parameter is ignored.
If the wait time is 0, an entry is placed on the data queue when the send request completes. The application should then perform a Receive Operation Completion (QNMRCVOC) to obtain the results of the send operation. It is recommended that a minimum of 30 seconds be used for the wait time. Larger values may be required in some networks.

The current value for the mode name of the application identified by the handle determines which session is used to send the request.

**Authorities and Locks**

None.

**Required Parameter Group**

**Handle**

INPUT; BINARY(4)

The unique identifier for this application, which was returned by the Start Application (QNMSTRAP) API.

**Remote application name**

INPUT; CHAR(24)

The name of the remote application to which the request is sent. The first 8 characters contain the network ID, the second 8 characters contain the control point name (or the logical unit name may be used), and the third 8 characters contain the application name of the remote application.

**Request identifier**

OUTPUT; CHAR(53)

The identifier that the system assigned to this request. This identifier must be kept by the application to receive replies to the request (if any).

**Send buffer**

INPUT; CHAR(*)

The data record being sent.

**Length of send buffer**

INPUT; BINARY(4)

The size of the data record being sent. The send buffer can range in size from 0 through 31739.

**Request type**

INPUT; CHAR(10)

The type of data to be sent:

*RQS This is a request only; no reply is expected.
*RQSRPY A reply is expected to this request.

**Post reply**

INPUT; CHAR(10)

Whether entries should be put on the data queue when replies to this request arrive.

*NO Do not place a reply entry on a data queue.
*YES Place the reply entry on the data queue associated with the handle.

**Wait time**

INPUT; BINARY(4)
The amount of time the application waits for the send operation to complete.

-1  Waits for the send operation to complete or for a condition such that the operation cannot complete (for example, a communications failure).
0   Does not wait for the operation to complete.
1-99999 The number of seconds the application waits.

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

<table>
<thead>
<tr>
<th>Message ID</th>
<th>Error Message Text</th>
</tr>
</thead>
<tbody>
<tr>
<td>CPF24B4 E</td>
<td>Severe error while addressing parameter list.</td>
</tr>
<tr>
<td>CPF3C90 E</td>
<td>Literal value cannot be changed.</td>
</tr>
<tr>
<td>CPF3CF1 E</td>
<td>Error code parameter not valid.</td>
</tr>
<tr>
<td>CPF7AC0 E</td>
<td>&amp;3.&amp;4 cannot receive data at this time.</td>
</tr>
<tr>
<td>CPF7ADC E</td>
<td>Internal processing error.</td>
</tr>
<tr>
<td>CPF7ADD E</td>
<td>Operation did not complete.</td>
</tr>
<tr>
<td>CPF7ADF E</td>
<td>Session failure. Cannot send data at this time.</td>
</tr>
<tr>
<td>CPF7AD2 E</td>
<td>Send buffer length value &amp;3 not valid.</td>
</tr>
<tr>
<td>CPF7AD6 E</td>
<td>Request type value &amp;3 not valid.</td>
</tr>
<tr>
<td>CPF7AD9 E</td>
<td>Post reply value &amp;3 not valid.</td>
</tr>
<tr>
<td>CPF7AEC E</td>
<td>Wait time &amp;3 not between -1 and 99999.</td>
</tr>
<tr>
<td>CPF7AE0 E</td>
<td>Function requested not supported by remote system.</td>
</tr>
<tr>
<td>CPF7AE1 E</td>
<td>Sign-on to control point &amp;3.&amp;4 denied.</td>
</tr>
<tr>
<td>CPF7AE2 E</td>
<td>Handle &amp;1 not found.</td>
</tr>
<tr>
<td>CPF7AE3 E</td>
<td>Management Services transport operation not permitted.</td>
</tr>
<tr>
<td>CPF7AE5 E</td>
<td>Control point &amp;3.&amp;4 not found.</td>
</tr>
<tr>
<td>CPF7AE6 E</td>
<td>Communication with control point &amp;3.&amp;4 failed.</td>
</tr>
<tr>
<td>CPF7AE7 E</td>
<td>Remote application program &amp;5 not found.</td>
</tr>
<tr>
<td>CPF9872 E</td>
<td>Program or service program &amp;1 in library &amp;2 ended. Reason code &amp;3.</td>
</tr>
</tbody>
</table>

API introduced: V2R2

Start Application (QNMSTRAP) API

Required Parameter Group:

<table>
<thead>
<tr>
<th></th>
<th>Handle</th>
<th>Output</th>
<th>Binary(4)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Local application name</td>
<td>Input</td>
<td>Char(8)</td>
</tr>
<tr>
<td>3</td>
<td>Qualified data queue name</td>
<td>Input</td>
<td>Char(20)</td>
</tr>
<tr>
<td>4</td>
<td>Error code</td>
<td>I/O</td>
<td>Char(*)</td>
</tr>
</tbody>
</table>

Default Public Authority: *USE
Threadsafe: No

The Start Application (QNMSTRAP) API, an SNA/Management Services Transport API, starts support for a network management application. The handle returned by this SNA/Management Services program must be used on all subsequent API calls to identify this application. The handle is unique to a specific job.
The Start Application API must be called before any other SNA/Management Services Transport APIs.

**Authorities and Locks**

*Data queue name*

*CHANGE

*Target data queue*

Other than *NONE

*Data queue library*

*USE

**Required Parameter Group**

*Handle*

OUTPUT; BINARY(4)

A variable that represents an instance of an application using some function.

*Local application name*

INPUT; CHAR(8)

The application name to be associated with the handle.

*Qualified data queue name*

INPUT; CHAR(20)

The data queue that indicates when asynchronous events occur, and enables an application to receive requests. The first 10 characters specify the data queue object name, and the last 10 characters specify the library name.

The data queue object must exist when this SNA/Management Services program is called. The maximum entry length of the data queue must be at least 80 bytes. You must specify FIFO for the sequence of the data when you create the data queue.

*NONE* A data queue is not used.

The following special values may be specified for the library name:

*LIBL* The library list.

*CURLIB* The job’s current library.

*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**

<table>
<thead>
<tr>
<th>Message ID</th>
<th>Error Message Text</th>
</tr>
</thead>
<tbody>
<tr>
<td>CPF24B4 E</td>
<td>Severe error while addressing parameter list.</td>
</tr>
<tr>
<td>CPF3C90 E</td>
<td>Literal value cannot be changed.</td>
</tr>
<tr>
<td>CPF3CF1 E</td>
<td>Error code parameter not valid.</td>
</tr>
<tr>
<td>CPF7ADA E</td>
<td>Data queue &amp;4/&amp;3 not valid.</td>
</tr>
<tr>
<td>CPF9800 E</td>
<td>All CPF98xx messages could be signaled. xx is from 01 to FF.</td>
</tr>
</tbody>
</table>

API introduced: V2R2
Alert APIs

The alert APIs let your application create alerts, notify the i5/OS(TM) alert manager of alerts that need to be handled, and allow you to retrieve alerts and alert data. The generate and send APIs differ from ordinary i5/OS alert processing in that they let your application create an alert at any time without sending an alertable message to an alertable message queue. (An alertable message queue is a message queue that has been created or changed with the allow alerts (ALWALR) parameter specified as yes.) The retrieve API allows your application, in conjunction with alert filtering, to perform user-defined actions based on the contents of the alert.

For more information on creating and sending i5/OS alerts, see the [Alerts Support book](#).

The alert APIs are:
- “Generate Alert (QALGENA) API” (QALGENA) creates an alert for a particular message ID.
- “Retrieve Alert (QALRTVA) API” on page 39 (QALRTVA) retrieves an alert from the alert database.
- “Send Alert (QALSNDA) API” on page 43 (QALSNDA) sends an alert to the i5/OS alert manager for processing.

Generate Alert (QALGENA) API

Required Parameter Group:

<table>
<thead>
<tr>
<th>Parameter Group</th>
<th>Required Parameter</th>
<th>Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Receiver variable</td>
<td>Output</td>
</tr>
<tr>
<td>2</td>
<td>Length of receiver variable</td>
<td>Input</td>
</tr>
<tr>
<td>3</td>
<td>Size of alert</td>
<td>Output</td>
</tr>
<tr>
<td>4</td>
<td>Qualified alert table (message file) name</td>
<td>Input</td>
</tr>
<tr>
<td>5</td>
<td>Message ID</td>
<td>Input</td>
</tr>
<tr>
<td>6</td>
<td>Message data</td>
<td>Input</td>
</tr>
<tr>
<td>7</td>
<td>Length of message data</td>
<td>Input</td>
</tr>
<tr>
<td>8</td>
<td>Error code</td>
<td>I/O</td>
</tr>
</tbody>
</table>

Default Public Authority: *USE
Threadsafe: No

The Generate Alert (QALGENA) API, an alert API, creates an alert for a particular message ID. The alert is then returned to the calling program.

Authorities and Locks

*Alert Table Authority*
- *USE

*Message File Authority*
- *READ

*Library Authority*
- *USE

Required Parameter Group

Receiver variable (alert major vector)
- OUTPUT; CHAR(*)
The variable that receives the generated alert, in the format of an SNA alert major vector. This area must be large enough to hold the alert. If the area is too small, the alert is not returned. A suggested size is 512 bytes because alerts cannot exceed 512 bytes for the i5/OS licensed program.

**Length of receiver variable**
```plaintext```
INPUT; BINARY(4)
```

The length of the receiver variable parameter. If this is too small, no data is returned, and the API ends with an exception.

**Size of alert**
```plaintext```
OUTPUT; BINARY(4)
```

The size of the successfully generated alerts. If the size of the alert is smaller than the length of receiver variable, the alert is returned in the receiver variable. If the size of the alert is greater than the length of receiver variable, the alert is not returned.

**Qualified alert table (message file) name**
```plaintext```
INPUT; CHAR(20)
```

The name of the alert table where the alert description defining the alert is stored, and the name of the library in which it resides. This parameter also identifies the name of the message file to use because the alert table and message file must have the same name.

The first 10 characters contain the alert table name, and the second 10 characters contain the name of the library.

Valid values for the library name are:

- `*CURLIB` The job's current library. The alert table and message file must both be in this library.
- `*LIBL` The library list. The alert table and message file can be in different libraries, but the libraries must both be in the library list.
- `Specific library` The alert table and message file must be in the same library.

**Message ID**
```plaintext```
INPUT; CHAR(7)
```

The message ID of the alert description defining the alert. This parameter also identifies the message ID of the corresponding message description. The message does not need to be an alertable message. The ALROPT parameter of the Display Message Description (DSPMSGD) command is ignored and an alert is always returned, provided that an alert description with the given message ID exists in the given alert table.

**Message data**
```plaintext```
INPUT; CHAR(*)
```

The message data to use for message substitution on the alert description and message description. The format of the message data is the same as used for the Send Program Message (SNDPGMMSG) command and the Send Program Message (QMHSNDPM) API. See the [Programming] topic for more information about message data in general and the online help for details about the SNDPGMMSG command.

**Length of message data**
```plaintext```
INPUT; BINARY(4)
```

The length of the message data provided by the message data parameter above.

**Error code**
```plaintext```
I/O; CHAR(*)
```

The structure in which to return error information. For the format of the structure, see [Error Code Parameter]
Error Handling
Some of the QALGENA API errors allow an alert to be returned to the application. An alert is returned to the application in these cases:
• The user is not authorized to the message file.
• A message cannot be added to the alert because the message or message file cannot be found.

No alert is returned to the application in these cases:
• The user is not authorized to the alert table.
• i5/OS alert support cannot build an alert because no message ID is given, the alert table is missing or damaged, or no alert description is found. In this case, the alert generated could only be a cause-undetermined alert of little or no value.

Error Messages

<table>
<thead>
<tr>
<th>Message ID</th>
<th>Error Message Text</th>
</tr>
</thead>
<tbody>
<tr>
<td>CPF24B4 E</td>
<td>Severe error while addressing parameter list.</td>
</tr>
<tr>
<td>CPF3C90 E</td>
<td>Literal value cannot be changed.</td>
</tr>
<tr>
<td>CPF3CF1 E</td>
<td>Error code parameter not valid.</td>
</tr>
<tr>
<td>CPF7B01 E</td>
<td>Receiver variable too small to hold alert.</td>
</tr>
<tr>
<td>CPF7B02 E</td>
<td>Message ID not valid.</td>
</tr>
<tr>
<td>CPF7B03 E</td>
<td>Alert table &amp;2/&amp;1 not found.</td>
</tr>
<tr>
<td>CPF7B04 E</td>
<td>Alert description &amp;1 not found in alert table &amp;4/&amp;3.</td>
</tr>
<tr>
<td>CPF7B05 E</td>
<td>Message file &amp;2/&amp;1 not available for alert processing.</td>
</tr>
<tr>
<td>CPF7B06 E</td>
<td>Message &amp;1 not found in message file &amp;4/&amp;3 for alert processing.</td>
</tr>
<tr>
<td>CPF7B10 E</td>
<td>Length parameter &amp;1 is not valid.</td>
</tr>
<tr>
<td>CPF9802 E</td>
<td>Not authorized to object &amp;2 in &amp;3.</td>
</tr>
<tr>
<td>CPF9872 E</td>
<td>Program or service program &amp;1 in library &amp;2 ended. Reason code &amp;3.</td>
</tr>
</tbody>
</table>

API introduced: V2R1

Retrieve Alert (QALRTVA) API

Required Parameter Group:

| 1 | Receiver variable | Output | Char(*) |
| 2 | Length of receiver variable | Input | Binary(4) |
| 3 | Format | Input | Char(8) |
| 4 | Alert log identifier | Input | Char(8) |
| 5 | Error code | I/O | Char(*) |

Default Public Authority: *USE
Threadsafe: No

The Retrieve Alert (QALRTVA) API, an alert API, retrieves an alert from the alert database. Different formats of data can be returned depending on what value is specified for the format parameter.

This API can be used to automate alerts such that an alert notification is sent to a data queue monitored by a user application whenever an alert is processed or received.

Authorities and Locks
None.
**Required Parameter Group**

**Receiver variable**

OUTPUT; CHAR(*)

The area to receive the formatted data. The data is formatted based on the format parameter. The data is either part of the alert or the alert major vector itself.

**Length of receiver variable**

INPUT; BINARY(4)

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

**Format**

INPUT; CHAR(8)

The format of the data to return. The possible formats are:

- **ALRT0100**
  
  Does not return the alert. It returns information about the alert that can be seen on the Work with Alerts display. See “ALRT0100 Format” for more information.

- **ALRT0200**
  
  Does return the alert and some information about the alert that is not contained within the alert. See “ALRT0200 Format” on page 41 for more information.

**Alert log identifier**

INPUT; CHAR(8)

The identifier used to retrieve the alert from the alert log. The log ID can be found in the alert notification record, which is displayed below after the ALRT0100 and ALRT0200 Format tables. The notification record is placed on a data queue during alert filtering by the send to data queue action.

**Error code**

I/O; CHAR(*)

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

**ALRT0100 Format**

See “Field Descriptions” on page 41 for descriptions of the fields in this format.

<table>
<thead>
<tr>
<th>Offset</th>
<th>Type</th>
<th>Field</th>
</tr>
</thead>
<tbody>
<tr>
<td>Dec</td>
<td>Hex</td>
<td>Byte available</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>Bytes available</td>
</tr>
<tr>
<td>4</td>
<td>4</td>
<td>Bytes returned</td>
</tr>
<tr>
<td>8</td>
<td>8</td>
<td>Timestamp</td>
</tr>
<tr>
<td>16</td>
<td>10</td>
<td>User assigned to this alert</td>
</tr>
<tr>
<td>26</td>
<td>1A</td>
<td>Group assigned to this alert</td>
</tr>
<tr>
<td>36</td>
<td>24</td>
<td>Filter used</td>
</tr>
<tr>
<td>56</td>
<td>38</td>
<td>Alert ID</td>
</tr>
<tr>
<td>60</td>
<td>3C</td>
<td>Problem ID</td>
</tr>
<tr>
<td>70</td>
<td>46</td>
<td>Origin system of problem</td>
</tr>
<tr>
<td>90</td>
<td>5A</td>
<td>Alert holding flag</td>
</tr>
<tr>
<td>91</td>
<td>5B</td>
<td>Local or received alert flag</td>
</tr>
</tbody>
</table>
### ALRT0200 Format

See "Field Descriptions" for descriptions of the fields in this format.

<table>
<thead>
<tr>
<th>Offset</th>
<th>Dec</th>
<th>Hex</th>
<th>Type</th>
<th>Field</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>0</td>
<td>0</td>
<td>BINARY(4)</td>
<td>Bytes available</td>
</tr>
<tr>
<td></td>
<td>4</td>
<td>4</td>
<td>BINARY(4)</td>
<td>Bytes returned</td>
</tr>
<tr>
<td></td>
<td>8</td>
<td>8</td>
<td>CHAR(8)</td>
<td>Timestamp</td>
</tr>
<tr>
<td></td>
<td>16</td>
<td>10</td>
<td>CHAR(10)</td>
<td>User assigned to this alert</td>
</tr>
<tr>
<td></td>
<td>26</td>
<td>1A</td>
<td>CHAR(10)</td>
<td>Group assigned to this alert</td>
</tr>
<tr>
<td></td>
<td>36</td>
<td>24</td>
<td>CHAR(20)</td>
<td>Filter used</td>
</tr>
<tr>
<td></td>
<td>56</td>
<td>38</td>
<td>CHAR(512)</td>
<td>Alert major vector</td>
</tr>
</tbody>
</table>

### Alert Notification Record

<table>
<thead>
<tr>
<th>Offset</th>
<th>Dec</th>
<th>Hex</th>
<th>Type</th>
<th>Field</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>0</td>
<td>0</td>
<td>CHAR(10)</td>
<td>Alert Notification (always ‘ALRFTR’)</td>
</tr>
<tr>
<td></td>
<td>10</td>
<td>A</td>
<td>CHAR(2)</td>
<td>Function code (always ‘01’ = Alert processed)</td>
</tr>
<tr>
<td></td>
<td>12</td>
<td>C</td>
<td>CHAR(10)</td>
<td>Group assigned to this alert</td>
</tr>
<tr>
<td></td>
<td>22</td>
<td>16</td>
<td>CHAR(10)</td>
<td>Filter used</td>
</tr>
<tr>
<td></td>
<td>32</td>
<td>20</td>
<td>CHAR(10)</td>
<td>Library of the filter</td>
</tr>
<tr>
<td></td>
<td>42</td>
<td>2A</td>
<td>CHAR(8)</td>
<td>Alert log identifier</td>
</tr>
<tr>
<td></td>
<td>50</td>
<td>32</td>
<td>CHAR(30)</td>
<td>Reserved</td>
</tr>
</tbody>
</table>

### Field Descriptions

**Alert description code point.** The description of the alert. The text is found in the QALRMSG message file in the QSYS library. The prefix for the message ID is ALD, and the suffix is the value of this field.
Alert held flag. If the alert has ever been held for the purpose of sending to the focal point, this flag is set to 1; if the alert has never been held, it is set to 0.

Alert holding flag. If the alert is currently being held to send to the focal point system, this flag is set to H; if not, the field is blank.

Alert ID. An assigned value for a particular alert.

Alert major vector. This is the SNA alert major vector.

Alert type code point. The type of alert. The text for the code point is found in the QALRMSG message file in the QSYS library. The prefix for the message ID is ALT, and the suffix is the value of this field followed by 00.

Analysis available flag. If further problem analysis is available for this problem or if the alert is for a problem analysis message, then this flag is set to 1; if the message is not for problem analysis, it is set to 0.

Bytes available. The number of bytes of data available to be returned.

Bytes returned. The number of bytes of data that were returned.

Delayed alert flag. If the alert was ever delayed, this flag is set to 1; if it has never been delayed, it is set to 0.

Filter used. The filter used to categorize the alert when it was first processed. The filter is not dynamically updated.

First probable cause code point. The most probable cause for an alert. The text for the description is found in the QALRMSG message file in the QSYS library. The prefix for the message ID is ALP, and the suffix is the value of this field.

Group assigned to this alert. The group the alert is assigned to when the alert is first filtered in the system. This value can be changed from the Work with Alerts display.

Local or received alert flag. If the alert is a locally generated alert, this flag is set to L; if it is a received alert, the flag is set to R.

Operator-generated alert flag. If the alert was generated by an operator, this flag is set to 1; if not, it is set to 0.

Origin system of problem. The system that was the origin of the associated problem entry. If no problem log entry is associated with the alert, this field is blank.

Problem ID. The ID of the problem associated with the alert. If no problem log entry is associated with the alert, this field is blank.

Resource name. The name of the resource that detected the error condition. The resource name indicates the location of the actual resource with the problem that created the alert. Generally, this is the control point name of the origin system.

Resource type. The type of resource that detected the error condition, for example, diskette, tape, printer, line, or display. The failing resource is the lowest resource in the resource hierarchy. See the [Alerts Support](#) book for the values of the resource type.
**Timestamp.** The machine timestamp (time of day) the alert was logged.

**User assigned to this alert.** The user the alert is assigned to when the alert is first filtered into the system. This value can be changed from the Work with Alerts display.

### Error Messages

<table>
<thead>
<tr>
<th>Message ID</th>
<th>Error Message Text</th>
</tr>
</thead>
<tbody>
<tr>
<td>CPF24B4 E</td>
<td>Severe error while addressing parameter list.</td>
</tr>
<tr>
<td>CPF3CF1 E</td>
<td>Error code parameter not valid.</td>
</tr>
<tr>
<td>CPF3C21 E</td>
<td>Format name &amp;1 is not valid.</td>
</tr>
<tr>
<td>CPF3C90 E</td>
<td>Literal value cannot be changed.</td>
</tr>
<tr>
<td>CPF7B10 E</td>
<td>Length parameter &amp;1 is not valid.</td>
</tr>
<tr>
<td>CPF7B11 E</td>
<td>Alert not found.</td>
</tr>
<tr>
<td>CPF9845 E</td>
<td>Error occurred while opening file &amp;1.</td>
</tr>
<tr>
<td>CPF9846 E</td>
<td>Error while processing file &amp;1 in library &amp;2.</td>
</tr>
<tr>
<td>CPF9847 E</td>
<td>Error occurred while closing file &amp;1 in library &amp;2.</td>
</tr>
<tr>
<td>CPF9872 E</td>
<td>Program or service program &amp;1 in library &amp;2 ended. Reason code &amp;3.</td>
</tr>
</tbody>
</table>

API introduced: V2R2

---

### Send Alert (QALSNDNA) API

**Required Parameter Group:**

<table>
<thead>
<tr>
<th></th>
<th>Alert major vector</th>
<th>Input</th>
<th>Char(*)</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>Length of alert major vector</td>
<td>Input</td>
<td>Binary(4)</td>
</tr>
<tr>
<td>3</td>
<td>Local or received indicator</td>
<td>Input</td>
<td>Char(1)</td>
</tr>
<tr>
<td>4</td>
<td>Origin</td>
<td>Input</td>
<td>Char(10)</td>
</tr>
<tr>
<td>5</td>
<td>Error code</td>
<td>I/O</td>
<td>Char(*)</td>
</tr>
</tbody>
</table>

Default Public Authority: *USE

Threadsafe: No

The Send Alert (QALSNDNA) API, an alert API, sends an alert to the i5/OS alert manager for processing. The alert is created by calling the Generate Alert (QALGENA) API. An alert may be received by your application from another system or it can be built by other means.

When the i5/OS alert manager receives the alert, it handles it like any other alert on the system. The alert function is notified of the alert, and the alert can be logged and forwarded to a focal point or central site. The alert can be treated as either a locally generated alert or a received alert. The i5/OS alert manager updates the hierarchical information of received alerts with the name of the iSeries server control point that is handling the alert (that is, the LCLCPNAME network attribute value).

### Authorities and Locks

None.

### Required Parameter Group

**Alert major vector**

```
INPUT; CHAR(*)
```

The variable that contains the alert major vector.
Length of alert major vector
  INPUT; BINARY(4)

The length of the alert, in bytes.

Valid values are 1 through 512.

Local or received indicator
  INPUT; CHAR(1)

One of these values, indicating whether the alert is handled as locally generated or received:

L  Locally generated alert. This alert is listed in the output from the Work with Alerts (WRKALR) command using the display option (DSPOPT) parameter with the *LOCAL special value. The alert hierarchy is not changed to add the current system’s name.
R  Received alert. This alert is listed in the output from the Work with Alerts (WRKALR) command using the display options (DSPOPT) parameter with the *RCV special value. The system name is added to the processing node list. The current system’s name, stored in the LCLCPNAME network attribute, is added to the alert hierarchy.

Origin
  INPUT; CHAR(10)

The origin of the alert. This value is not included in the alert. It is used only in the substitution text for messages CPI7B62 (Alert received from &1) and CPI7B60 (Incorrect alert received from &1), which are sent to the QSYSOPR message queue. Thus, you could use it for the name of the program generating a locally generated alert, or the name of the system sending a received alert.

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

<table>
<thead>
<tr>
<th>Message ID</th>
<th>Error Message Text</th>
</tr>
</thead>
<tbody>
<tr>
<td>CPF24B4 E</td>
<td>Severe error while addressing parameter list.</td>
</tr>
<tr>
<td>CPF3C90 E</td>
<td>Literal value cannot be changed.</td>
</tr>
<tr>
<td>CPF3CF1 E</td>
<td>Error code parameter not valid.</td>
</tr>
<tr>
<td>CPF7B07 E</td>
<td>Alert exceeds maximum size allowed.</td>
</tr>
<tr>
<td>CPF7B08 E</td>
<td>Alert is not valid.</td>
</tr>
<tr>
<td>CPF7B09 E</td>
<td>Value specified for parameter &amp;1 not valid.</td>
</tr>
<tr>
<td>CPF7B10 E</td>
<td>Length parameter &amp;1 is not valid.</td>
</tr>
<tr>
<td>CPF9872 E</td>
<td>Program or service program &amp;1 in library &amp;2 ended. Reason code &amp;3.</td>
</tr>
</tbody>
</table>

API introduced: V2R1

---

Node List API

A node list contains a list of systems identified by a remote location name and an address type. Nodes with an SNA address type are identified by a network ID and a control point name. Nodes with an internet protocol (IP) type are identified by a host name or an internet address.

The node list API is:

- "List Node List Entries (QFVLSTNL) API" on page 45 (QFVLSTNL) returns, in a user space, a list of the nodes contained in the specified node list object.
List Node List Entries (QFVLSTNL) API

Required Parameter Group:

<table>
<thead>
<tr>
<th>Required Parameter Group</th>
<th>Input</th>
<th>Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>Qualified user space name</td>
<td>Input</td>
<td>Char(20)</td>
</tr>
<tr>
<td>Format name</td>
<td>Input</td>
<td>Char(8)</td>
</tr>
<tr>
<td>Qualified node list name</td>
<td>Input</td>
<td>Char(20)</td>
</tr>
<tr>
<td>Error code</td>
<td>I/O</td>
<td>Char(*)</td>
</tr>
</tbody>
</table>

Default Public Authority: *USE
Threadsafe: No

The List Node List Entries (QFVLSTNL) API returns, in a user space, a list of the nodes contained in the specified node list object.

Authorities and Locks

Node List Object Authority
*USE

Node List Object Lock
*SHRRD

User Space Authority
*CHANGE

User Space Library Authority
*EXECUTE

User Space Lock
*EXCLRD

Required Parameter Group

Qualified user space name
*INPUT; CHAR (20)

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

The following special values can be used for the library name:

*CURLIB The current library.
*LBL The library list.

Format name
*INPUT; CHAR(8)

The format of the data placed in the user space. The valid value is:

NODL0100 All node list entry information returned. This format is explained in "NODL0100 List Data Section" on page 47.

Qualified node list name
*INPUT; CHAR(20)
The name of the node list object from which the entries are to be retrieved. The first 10 characters contain the node list name. The second 10 characters contain the name of the library where the node list is located.

Special values for the name of the node list library are:

* CURLIB  The current library.
* LIBL  The library list.

**Error code**

I/O; CHAR(*)

The structure in which to return error information. For the format of the structure, see [Error Code Parameter](#).

**Format of the Generated Lists**

The returned user space will contain:

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

For details about the user area and generic header, see [User Space Format for List APIs](#). For details about the other items, see the following sections. For a detailed description of each field in the information returned, see [“Field Descriptions” on page 47](#).

**Input Parameter Section**

<table>
<thead>
<tr>
<th>Offset</th>
<th>Type</th>
<th>Field</th>
</tr>
</thead>
<tbody>
<tr>
<td>Dec</td>
<td>Hex</td>
<td>CHAR(10)</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>User space name specified</td>
</tr>
<tr>
<td>10</td>
<td>A</td>
<td>User space library name specified</td>
</tr>
<tr>
<td>20</td>
<td>14</td>
<td>Format name specified</td>
</tr>
<tr>
<td>28</td>
<td>1C</td>
<td>Node list object name specified</td>
</tr>
<tr>
<td>38</td>
<td>26</td>
<td>Node list object library name specified</td>
</tr>
</tbody>
</table>

**Header Section**

<table>
<thead>
<tr>
<th>Offset</th>
<th>Type</th>
<th>Field</th>
</tr>
</thead>
<tbody>
<tr>
<td>Dec</td>
<td>Hex</td>
<td>CHAR(10)</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>User space name used</td>
</tr>
<tr>
<td>10</td>
<td>A</td>
<td>User space library name used</td>
</tr>
<tr>
<td>20</td>
<td>14</td>
<td>Node list object name used</td>
</tr>
<tr>
<td>30</td>
<td>1E</td>
<td>Node list object library name used</td>
</tr>
</tbody>
</table>
NODL0100 List Data Section

<table>
<thead>
<tr>
<th>Offset</th>
<th>Dec</th>
<th>Hex</th>
<th>Type</th>
<th>Field</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>CHAR(1)</td>
<td>Address type</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>CHAR(20)</td>
<td>Remote location name (for address type 1)</td>
</tr>
<tr>
<td>21</td>
<td>15</td>
<td>47</td>
<td>CHAR(50)</td>
<td>Text description</td>
</tr>
<tr>
<td>71</td>
<td>47</td>
<td>48</td>
<td>CHAR(1)</td>
<td>Reserved</td>
</tr>
<tr>
<td>72</td>
<td>48</td>
<td>4C</td>
<td>BINARY(4)</td>
<td>Text description CCSID</td>
</tr>
<tr>
<td>76</td>
<td>4C</td>
<td>76</td>
<td>CHAR(256)</td>
<td>Remote location name (for address type 1, 2, or 3)</td>
</tr>
</tbody>
</table>

Field Descriptions

Address type. The type of address returned. The following values are used:

1. SNA APPN address type. The network ID and control point name are given in both the remote location name field at offset 1 and at offset 76.
2. Internet protocol (IP) address type. The host name is given in the remote location name field at offset 76.
3. Internet protocol (IP) address type. The internet address is given in the remote location name field at offset 76.

Format name specified. The format name specified as input for the API.

Node list object library name specified. The name of the node list object library as specified on the call to the API.

Node list object library name used. The actual name of the node list object library used to report data.

Node list object name specified. The node list object name as specified on the call to the API. Nodes with an SNA address type are identified by a network ID and a control point name. Nodes with an internet protocol (IP) type are identified by a host name or an internet address. The system-recognized identifier for the object type is *NODL.

Node list object name used. The actual node list object name used to report data. The system-recognized identifier for the object type is *NODL.

Remote location name (for address type x). The name of a system in a network.

The remote location name field at offset 1 contains an 8-character SNA network ID, an 8-character control point name, and 4 reserved characters. The remote location name field at offset 1 is only used if the address type is 1 (SNA). Otherwise, the field at offset 1 is blank.

The contents of the remote location name field at offset 76 depends on the address type:

1. Contains the same name as the field at offset 1: an 8-character SNA network ID, an 8-character control point name, and 4 reserved characters.
2. Contains a 255-character host name and 1 reserved character.
3. Contains a 15-character internet address and 241 reserved characters.

Reserved. An ignored field.

Text description. The text description of the node list entry.
Text description CCSID. The coded character set identifier (CCSID) used for the text description.

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

User space library name used. The actual user space library name used to report data.

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

User space name used. The actual user space name used to report data.

Error Messages

<table>
<thead>
<tr>
<th>Message ID</th>
<th>Error Message Text</th>
</tr>
</thead>
<tbody>
<tr>
<td>CPF24B4 E</td>
<td>Severe error while addressing parameter list.</td>
</tr>
<tr>
<td>CPF3C21 E</td>
<td>Format name &amp;1 is not valid.</td>
</tr>
<tr>
<td>CPF3C90 E</td>
<td>Literal value cannot be changed.</td>
</tr>
<tr>
<td>CPF3CF1 E</td>
<td>Error code parameter not valid.</td>
</tr>
<tr>
<td>CPF813E E</td>
<td>Node list &amp;4 in &amp;9 damaged.</td>
</tr>
<tr>
<td>CPF9801 E</td>
<td>Object &amp;2 in library &amp;3 not found.</td>
</tr>
<tr>
<td>CPF9802 E</td>
<td>Not authorized to object &amp;2 in &amp;3.</td>
</tr>
<tr>
<td>CPF9803 E</td>
<td>Cannot allocate object &amp;2 in library &amp;3.</td>
</tr>
<tr>
<td>CPF9807 E</td>
<td>One or more libraries in library list deleted.</td>
</tr>
<tr>
<td>CPF9808 E</td>
<td>Cannot allocate one or more libraries on library list.</td>
</tr>
<tr>
<td>CPF9810 E</td>
<td>Library &amp;1 not found.</td>
</tr>
<tr>
<td>CPF9820 E</td>
<td>Not authorized to use library &amp;1.</td>
</tr>
<tr>
<td>CPF9830 E</td>
<td>Cannot assign library &amp;1.</td>
</tr>
<tr>
<td>CPF9838 E</td>
<td>User profile storage limit exceeded.</td>
</tr>
<tr>
<td>CPF9872 E</td>
<td>Program or service program &amp;1 in library &amp;2 ended. Reason code &amp;3.</td>
</tr>
</tbody>
</table>

API introduced: V2R3

Registered Filter APIs

A filter is a function you can use to assign events into groups and to specify actions to take for each group. The registered filter APIs allow a product to register a filter with the operating system. The product can receive notification of events recorded in a data queue by using the Send to Data Queue (SNDDTAQ) action of the Work with Filter Action Entry (WRKFTRACNE) command.

A user filter is the filter defined by the network attributes for alert filtering and by the system value for problem log filtering. A user filter and a registered filter differ in their function and their notification record. There can only be one user filter active at one time for each type of filter, but there can be multiple registered filters active at one time. All actions are active for a user filter, but only the SNDDTAQ action is active for a registered filter.

A product can use registered filter APIs for the following purposes:

• To register multiple filters at the same time for each event type (alert or problem log)
• To deregister a filter when notifications from that filter are no longer necessary
• To retrieve all the filters that are registered

The event notification record for a registered filter differs from notification records for other types of filters. The registered notification contains a common header for all events, as well as specific information based on the type of event. The common header includes the name of the notification, a function type, a
format, the filter name and library, the group name, and a timestamp. The specific information for the problem log includes the problem ID, the last event logged, and the timestamp for the last event.

The registered filter APIs are:

- "Deregister Filter Notifications (QNMDRGFN) API" (QNMDRGFN) deregisters a filter. If the identified filter is not currently registered, an error is returned.
- "Register Filter Notifications (QNMRFN) API" on page 50 (QNMRFN) registers a filter to send notifications for a specific event to a data queue.
- "Retrieve Registered Filters (QNMRRGF) API" on page 53 (QNMRRGF) returns all filters registered for a filter type.

Deregister Filter Notifications (QNMDRGFN) API

Required Parameter Group:

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 Qualified filter name</td>
<td>Input, Char(20)</td>
</tr>
<tr>
<td>2 Filter type</td>
<td>Input, Char(10)</td>
</tr>
<tr>
<td>3 Error code</td>
<td>I/O, Char(*)</td>
</tr>
</tbody>
</table>

Default Public Authority: *USE
Threadsafe: No

The Deregister Filter Notifications (QNMDRGFN) API deregisters a filter. If the identified filter is not currently registered, an error is returned.

Authorities and Locks

Registered Filter
*USE

Registered Filter Library
*USE

Required Parameter Group

Qualified filter name
INPUT; CHAR(20)

The qualified filter name that is being deregistered. The first 10 characters are the filter name, and the last 10 characters are the library name. Special values *LIBL and *CURLIB are not supported for the library name.

Filter type
INPUT; CHAR(10)

The filter type that is being deregistered.

Special values supported are:

*ALR Alert filter
*PRB Problem log filter

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

<table>
<thead>
<tr>
<th>Message ID</th>
<th>Error Message Text</th>
</tr>
</thead>
<tbody>
<tr>
<td>CPF24B4 E</td>
<td>Severe error while addressing parameter list.</td>
</tr>
<tr>
<td>CPF3C90 E</td>
<td>Literal value cannot be changed.</td>
</tr>
<tr>
<td>CPF3CF1 E</td>
<td>Error code parameter not valid.</td>
</tr>
<tr>
<td>CPF8100 E</td>
<td>All CPF81xx messages could be returned. xx is from 01 to FF.</td>
</tr>
<tr>
<td>CPF91D1 E</td>
<td>Filter &amp;1/&amp;2 of type &amp;3 was not deregistered.</td>
</tr>
<tr>
<td>CPF91D2 E</td>
<td>Filter type &amp;3 not correct for this operation.</td>
</tr>
<tr>
<td>CPF91D4 E</td>
<td>Filter &amp;1/&amp;2 of type &amp;3 is not registered.</td>
</tr>
<tr>
<td>CPF9800 E</td>
<td>All CPF98xx messages could be signaled. xx is from 01 to FF.</td>
</tr>
</tbody>
</table>

API introduced: V2R1

---

Register Filter Notifications (QNMRGFN) API

Required Parameter Group:

1. Qualified filter name (Input) Char(20)
2. Filter type (Input) Char(10)
3. Error code (I/O) Char(*)

Default Public Authority: *USE

Threadsafe: No

The Register Filter Notifications (QNMRGFN) API registers a filter to send notifications for a specific event to a data queue. This data queue is defined in the *SNDDTAQ action entry field of the registered filter. These notification records are the asynchronous output of this API. All CL filter commands can be performed against a registered filter.

A filter can only be registered once. If a registered filter is being registered again, an error is returned. No checking is done to see if the registered filter is the same as another filter, such as an active user filter.

A check to prevent duplicate notifications for a single event is made when the registered filters are processed. If a notification record for a registered filter is a duplicate of another notification record (such as an active user filter) the registered filter is not processed.

If an error occurs in accessing a registered filter, or while enqueuing the notification record, the filter is automatically deregistered, and an informational message (CPI91D5) is sent to the system operator (QSYSOPR) message queue. This prevents the system from encountering the error again.

The registration for a filter remains active after the initial program load of a system. This ensures that a product receives all notifications. A product should register its filter either when the product is installed or when the product is started. A product should check to see that its filter is registered at its start-up time to ensure that its filter was not automatically deregistered. You do not have to deregister a filter to change the filter.

## Authorities and Locks

Registered Filter

*USE
Registered Filter Library

*USE

Required Parameter Group

Qualified filter name
input; char(20)

The qualified filter name that is being registered. The first 10 characters are the filter name, and the last 10 characters are the library name. Special values *LIBL and *CURLIB are not supported for the library name.

Filter type
input; char(10)

The type of filter. The following filter types are specified in the format field.

Special values supported are:

*ALR Alert filter. (See “Alert Filter” for details about the format.)
*PRB Problem log filter. (See “Problem Log Filter” on page 52 for details about the format.)

Error code
input; char(*)

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

Format of Registered Filter Data Queue Notification

See “Field Descriptions” on page 52 for descriptions of the fields in this format.

<table>
<thead>
<tr>
<th>Offset</th>
<th>Type</th>
<th>Field</th>
</tr>
</thead>
<tbody>
<tr>
<td>Dec</td>
<td>Hex</td>
<td>Dec</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>char(10) Notification name</td>
</tr>
<tr>
<td>10</td>
<td>A</td>
<td>char(2) Function code</td>
</tr>
<tr>
<td>12</td>
<td>C</td>
<td>char(8) Format</td>
</tr>
<tr>
<td>20</td>
<td>14</td>
<td>char(10) Filter name</td>
</tr>
<tr>
<td>30</td>
<td>1E</td>
<td>char(10) Filter library name</td>
</tr>
<tr>
<td>40</td>
<td>28</td>
<td>char(10) Group name</td>
</tr>
<tr>
<td>50</td>
<td>32</td>
<td>char(8) Timestamp</td>
</tr>
<tr>
<td>58</td>
<td>3A</td>
<td>char(22) Reserved</td>
</tr>
<tr>
<td>80</td>
<td>50</td>
<td>char(*) Event-specific notification data</td>
</tr>
</tbody>
</table>

The types of event-specific notification data used by the format field are described below.

Alert Filter

See “Field Descriptions” on page 52 for descriptions of the fields in this format.

<table>
<thead>
<tr>
<th>Offset</th>
<th>Type</th>
<th>Field</th>
</tr>
</thead>
<tbody>
<tr>
<td>Dec</td>
<td>Hex</td>
<td>Dec</td>
</tr>
<tr>
<td>80</td>
<td>50</td>
<td>char(512) Alert major vector</td>
</tr>
</tbody>
</table>
**Problem Log Filter**

See "Field Descriptions" for descriptions of the fields in this format.

<table>
<thead>
<tr>
<th>Offset</th>
<th>Type</th>
<th>Field</th>
</tr>
</thead>
<tbody>
<tr>
<td>80</td>
<td>CHAR(30)</td>
<td>Problem ID</td>
</tr>
<tr>
<td>110</td>
<td>CHAR(1)</td>
<td>Last event</td>
</tr>
<tr>
<td>111</td>
<td>CHAR(8)</td>
<td>Last event timestamp</td>
</tr>
</tbody>
</table>

**Field Descriptions**

**Alert major vector.** The actual alert that caused the notification.

**Event-specific notification data.** Data specific to the event identified in the function code. The data is in the format specified by the format variable.

**Filter library name.** The library that contains the filter.

**Filter name.** The name of the filter that sent the notification.

**Format.** The format of the event-specific data of a problem log filter. Valid values are:

- RGFN0100 Alert filter
- RGFN0200 Problem log filter

**Function code.** The event that caused the notification. Valid values are:

- 01 Alert event
- 02 Problem log event

**Group name.** The group into which the event was filtered.

**Last event.** The last event performed on the problem. Valid values are:

- 01 Problem entry opened
- 02 Service request received code
- 03 Opened by alert
- 10 Problem analyzed
- 11 Verification test ran
- 12 Recovery procedure ran
- 20 Prepared to report
- 21 Service request sent
- 22 Problem answered
- 23 Response sent
- 24 Reported by voice
- 25 Fixes transmitted
- 26 Change request submitted
- 27 Change request ended
- 30 Fix verified
- 41 Remote analysis
- 42 Remote verification ran
- 43 Remote recovery ran
Alert created
APAR created
APAR data collected
APAR data restored
APAR data deleted
Changed by CHGPRB
Deleted by DLTPRB
Recurring problem
Status changed
Status query sent
APAR data restored
APAR data deleted

Last event timestamp. The time at which the last event was performed.

Notification name. The type of notification record. It is set to *RGFN for all registered filter notifications.

Problem ID. The ID of the problem that caused the notification.

Reserved. An ignored field.

Timestamp. The time at which the event was processed.

Error Messages

<table>
<thead>
<tr>
<th>Message ID</th>
<th>Error Message Text</th>
</tr>
</thead>
<tbody>
<tr>
<td>CPF24B4 E</td>
<td>Severe error while addressing parameter list.</td>
</tr>
<tr>
<td>CPF3C7C E</td>
<td>User index is full.</td>
</tr>
<tr>
<td>CPF3C90 E</td>
<td>Literal value cannot be changed.</td>
</tr>
<tr>
<td>CPF3CF1 E</td>
<td>Error code parameter not valid.</td>
</tr>
<tr>
<td>CPF8100 E</td>
<td>All CPF81xx messages could be returned. xx is from 01 to FF.</td>
</tr>
<tr>
<td>CPF91D2 E</td>
<td>Filter type &amp;3 not correct for this operation.</td>
</tr>
<tr>
<td>CPF91D3 E</td>
<td>Filter &amp;1/&amp;2 with type &amp;3 is already registered.</td>
</tr>
<tr>
<td>CPF9800 E</td>
<td>All CPF98xx messages could be signaled. xx is from 01 to FF.</td>
</tr>
</tbody>
</table>

API introduced: V2R3

Retrieve Registered Filters (QNMRRGFF) API

Required Parameter Group:

<p>| | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Receiver variable</td>
<td>Output</td>
</tr>
<tr>
<td>2</td>
<td>Length of receiver variable</td>
<td>Input</td>
</tr>
<tr>
<td>3</td>
<td>Filter type</td>
<td>Input</td>
</tr>
<tr>
<td>4</td>
<td>Format</td>
<td>Input</td>
</tr>
</tbody>
</table>
The Retrieve Registered Filters (QNMRRGF) API returns all filters registered for a filter type. If not enough space exists to return all the registered filters, then as many registered filters as fit in the provided space are returned. A count of the total number of registered filters for the specific filter type is also returned.

**Authorities and Locks**

None.

**Required Parameter Group**

**Receiver variable**

OUTPUT; CHAR(*)

The variable in which the registered filters are returned. If the variable is not large enough to contain all the registered filters, then only as many registered filters will be returned as will fit in the receiver variable.

**Length of receiver variable**

INPUT; BINARY(4)

The length of the receiver variable parameter. If not enough space is given, then only the number of filters that fit will be returned. In this case the number of bytes available will be greater than the number of bytes returned.

**Filter type**

INPUT; CHAR(10)

The type of filter to be retrieved.

Special values supported are:

*ALR Alert filter
*PRB Problem log filter

**Format**

INPUT; CHAR(8)

The format of the receiver variable. The valid value is format RGFN0100. See **RGFN0100 Format** for the format of the receiver variable.

**Error code**

I/O; CHAR(*)

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

**RGFN0100 Format**

See **Field Descriptions** on page 55 for a description of the fields in this format.

<table>
<thead>
<tr>
<th>Offset</th>
<th>Dec</th>
<th>Hex</th>
<th>Type</th>
<th>Field</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>BINARY(4)</td>
<td>Bytes returned</td>
</tr>
<tr>
<td>4</td>
<td>4</td>
<td>4</td>
<td>BINARY(4)</td>
<td>Bytes available</td>
</tr>
</tbody>
</table>
### Field Descriptions

**Array of registered filters.** A 10-character filter name followed by a 10-character library name followed by a 10-character filter type.

**Bytes available.** The number of bytes of data that is available. If this number is greater than the bytes returned, the receiver variable was not large enough to contain all the registered filters.

**Bytes returned.** The number of bytes of data returned in the receiver variable.

**Element length.** The length of a single array element. For format RGFN0100 the length is 30 bytes.

**Filter library name.** The name of the library containing the registered filter.

**Filter name.** The name of the filter object that was registered.

**Filter type.** The type of the filter as defined by the FTRTYPE parameter on the Create Filter (CRTFTR) command.

**Returned registered filters.** The number of registered filters that is returned in the array.

**Total registered filters.** The total number of filters registered for the specified filter type.

### Error Messages

<table>
<thead>
<tr>
<th>Message ID</th>
<th>Error Message Text</th>
</tr>
</thead>
<tbody>
<tr>
<td>CPF24B4 E</td>
<td>Severe error while addressing parameter list.</td>
</tr>
<tr>
<td>CPF3CF1 E</td>
<td>Error code parameter not valid.</td>
</tr>
<tr>
<td>CPF3C21 E</td>
<td>Format name &amp;1 is not valid.</td>
</tr>
<tr>
<td>CPF3C24 E</td>
<td>Length of the receiver variable is not valid.</td>
</tr>
<tr>
<td>CPF3C90 E</td>
<td>Literal value cannot be changed.</td>
</tr>
<tr>
<td>CPF8100 E</td>
<td>All CPF81xx messages could be returned. xx is from 01 to FF.</td>
</tr>
<tr>
<td>CPF91D6 E</td>
<td>Filter type &amp;1 not valid.</td>
</tr>
<tr>
<td>CPF9800 E</td>
<td>All CPF98xx messages could be signaled. xx is from 01 to FF.</td>
</tr>
</tbody>
</table>

API introduced: V2R3
Change Request Management APIs

The change request management APIs add, remove, and list activities and retrieve change request descriptions.

The change request management APIs are:

- "Add Activity (QFVADDA) API“ (QFVADDA) adds an activity to the specified change request description.
- "List Activities (QFVLSTA) API” on page 61 (QFVLSTA) retrieves a list of activities from a qualified change request description.
- “Remove Activity (QFVRMVA) API” on page 67 (QFVRMVA) removes an activity from the specified change request description.
- "Retrieve Change Request Description (QFVRTVCD) API” on page 68 (QFVRTVCD) retrieves either general change request description information, or information pertaining to a single activity within the change request description.

Add Activity (QFVADDA) API

Required Parameter Group:

<table>
<thead>
<tr>
<th></th>
<th>Required Parameter Group</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Qualified change request description name</td>
<td>Input</td>
<td>Char(20)</td>
</tr>
<tr>
<td>2</td>
<td>Activity name</td>
<td>Input</td>
<td>Char(10)</td>
</tr>
<tr>
<td>3</td>
<td>Activity type</td>
<td>Input</td>
<td>Char(10)</td>
</tr>
<tr>
<td>4</td>
<td>Destination format</td>
<td>Input</td>
<td>Char(10)</td>
</tr>
<tr>
<td>5</td>
<td>Destination</td>
<td>Input</td>
<td>Array of Char(20)</td>
</tr>
<tr>
<td>6</td>
<td>Number of destinations</td>
<td>Input</td>
<td>Binary(4)</td>
</tr>
<tr>
<td>7</td>
<td>Condition list</td>
<td>Input</td>
<td>Array of Char(32)</td>
</tr>
<tr>
<td>8</td>
<td>Number of conditions</td>
<td>Input</td>
<td>Binary(4)</td>
</tr>
<tr>
<td>9</td>
<td>Start time</td>
<td>Input</td>
<td>Char(40)</td>
</tr>
<tr>
<td>10</td>
<td>Hold</td>
<td>Input</td>
<td>Char(10)</td>
</tr>
<tr>
<td>11</td>
<td>Function parameters</td>
<td>Input</td>
<td>Char(*)</td>
</tr>
<tr>
<td>12</td>
<td>Length of function parameters</td>
<td>Input</td>
<td>Binary(4)</td>
</tr>
<tr>
<td>13</td>
<td>Text</td>
<td>Input</td>
<td>Char(50)</td>
</tr>
<tr>
<td>14</td>
<td>Replace</td>
<td>Input</td>
<td>Char(10)</td>
</tr>
<tr>
<td>15</td>
<td>Activity added</td>
<td>Output</td>
<td>Char(10)</td>
</tr>
<tr>
<td>16</td>
<td>Error code</td>
<td>I/O</td>
<td>Char(*)</td>
</tr>
</tbody>
</table>

Default Public Authority: *USE
Threadsafe: No

The Add Activity (QFVADDA) API adds an activity to the specified change request description.

Authorities and Locks

Change request description authority
*CHANGE

Change request description lock
*EXCLRD

Required Parameter Group

Qualified change request description name
INPUT; CHAR(20)
The name and library of the qualified change request description to which an activity is added. The first ten characters contain the name of the change request description. The second ten characters contain the name of the library where the change request description is located.

The following special values can be used for library name:

*CURLIB The current library
*LIBL The library list

Activity name
INPUT; CHAR(10)

The name of the activity to add to the qualified change request description. The first character must be alphabetic or one of the special symbols $, @, or #. The remaining characters can be alphanumeric (A through Z, a through z, 0 through 9, and special symbols $, @, ., or _). The maximum number of activities allowed per change request description is 200.

Special values for the name of the activity are:

*GEN The activity name is generated automatically in the format QACTnnnnnn where nnnnnn is a multiple of 10 from 000010 to 999990.
*LAST This activity is the last activity to run for the change request. Only one activity in a change request may be specified with this value. The number of conditions must be 0. The start after time and date of start time parameter must be *CURRENT. The start before time and date of start parameter must be *ANY.

Activity type
INPUT; CHAR(10)

The name of the activity type to be added to the change request description. This value is defined by the user program. The API does not define the possible values. The characters that can be used for the activity type name are A through Z, 0 through 9, and *. Only IBM activity types should start with *.

Destination format
INPUT; CHAR(10)

Whether the activity runs on the local system, a node list name, or a list of 1 to 50 nodes specified by the user.

Special values for the destination format are:

*LCL The activity runs on the local system. The destination and number of destinations parameters are ignored if they are specified (the local network ID and control point name are assumed).
*NODL The activity runs on the systems identified in the node list name. The number of destinations parameter is ignored if it is specified. The node list is not restricted to 50 nodes.
*SNALST The activity runs on the systems identified in the list of user-specified SNA nodes.

Destination
INPUT; Array of CHAR(20)

The destination of the activity. The format of the destination is based on the destination format parameter.

When the destination format is *SNALST, the destination parameter is an array of the number of destinations elements.

The format for each element is:

Network ID CHAR(8)
Control point CHAR(8)
Reserved CHAR(4)
When the destination format is *NODL, the destination parameter is an array of one element with
the following format:

Node list name   CHAR(10)
Library          CHAR(10)

When the destination format is *LCL, the destination parameter is ignored.

**Number of destinations**

INPUT; BINARY(4)

The number of elements in the destination array. Valid values range from 1 through 50. The
number of destinations parameter is ignored if the destination format parameter is *LCL or
*NODL.

**Condition list**

INPUT: Array of CHAR(32)

The list of conditions that must be met before the activity can be run.

The format for each element is:

**Activity name**   CHAR(10)

The activity name which can be:

* activity name   the activity is conditioned on the activity name specified.
* generic name    the activity is conditioned on all the activities that match the generic name (partial
activity followed by an asterisk (*)).
* *PRV            the activity is conditioned on the activity that precedes it alphabetically.

**Relation**   CHAR(3)

Valid values are *EQ, *NE, *GT, *LT, *GE, and *LE.

**Reserved**   CHAR(3)

**Code**   BINARY(31)

The end code value to base conditioning on. Possible values are:

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0-99</td>
<td>user-specified</td>
</tr>
<tr>
<td>-1</td>
<td>(*SUCCESS) end code is any value from 0-9. If specified, relation must be *EQ or *NE.</td>
</tr>
<tr>
<td>-2</td>
<td>(*FAIL) end code is any value from 10-89. If specified, relation must be *EQ or *NE.</td>
</tr>
<tr>
<td>-3</td>
<td>(*NOTRUN) end code is any value from 90-99. If specified, relation must be *EQ or *NE.</td>
</tr>
<tr>
<td>-4</td>
<td>(*ANY) end code is any value from 0-99. If specified, relation must be *EQ.</td>
</tr>
</tbody>
</table>
Mode \hspace{1cm} \text{CHAR(10)}

Possible values are:

*ALLNODES
   all nodes of the conditioning activity must meet the completion criteria before this
   condition is considered met.

*SAMENODE
   nodes of the conditioned activity need only wait for the same node of the conditioning
   activity to meet the completion criteria before this condition is considered met.

Reserved \hspace{1cm} \text{Null}

Number of conditions
\text{INPUT; BINARY(4)}

The number of condition elements. The valid values are 0 through 5. When the number of
condition elements is 0, the activity specified is not dependent on any other activities to complete
before it can start, and the condition list parameter is ignored.

Start time
\text{INPUT; CHAR(40)}

The date and time window during which this activity can be started. The current date and time
values and next date values are determined when the change request is submitted. The structure is:

Start after time \hspace{1cm} \text{CHAR(10)}

The time after which this activity may be started. Special values supported are:

*CURRENT

Start after date \hspace{1cm} \text{CHAR(10)}

The date after which this activity may be started. Special value supported:

*CURRENT

*NEXT

Start before time \hspace{1cm} \text{CHAR(10)}

The time before which the activity must be started. If the activity cannot be started before this
time then it is never started. Special values supported are:

*ANY

*CURRENT

Start before date \hspace{1cm} \text{CHAR(10)}

The date before which the activity must be started. If the activity cannot be started before this date
then it is never started. Special values supported are:

*ANY

*CURRENT

*NEXT

The format for time is hhmmss. The format for date is cyymmd.

Hold \hspace{1cm} \text{INPUT; CHAR(10)}

Whether or not this activity is in Held status when it is submitted.
Valid values are *YES and *NO.

**Function parameters**

**INPUT; CHAR(*)**

The structure containing the specific function parameters. The format is known only by the program calling this API and the exit program that is called to display, change, print, or run this activity.

**Length of function parameters**

**INPUT; BINARY(4)**

The total length in bytes of the function parameters parameter. Valid values range from 1 through 12288 (12k).

**Text**

**INPUT; CHAR(50)**

The description of the activity.

**Replace**

**INPUT; CHAR(10)**

Whether an activity that already exists is replaced. If the activity name is found, it is replaced on this parameter.

*YES  The activity that already exists in the change request description is replaced.

*NO   The activity that already exists in the change request description is not replaced.

**Activity added**

**OUTPUT; CHAR(10)**

The name of the activity added to the change request description. This is useful when *GEN is specified for the activity name. This parameter is ignored if the activity already exists.

**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

<table>
<thead>
<tr>
<th>Message ID</th>
<th>Error Message Text</th>
</tr>
</thead>
<tbody>
<tr>
<td>CPF2150 E</td>
<td>Object information function failed.</td>
</tr>
<tr>
<td>CPF2151 E</td>
<td>Operation failed for &amp;2 in &amp;1 type *&amp;3.</td>
</tr>
<tr>
<td>CPF24B4 E</td>
<td>Severe error while addressing parameter list.</td>
</tr>
<tr>
<td>CPF3793 E</td>
<td>Machine storage limit reached.</td>
</tr>
<tr>
<td>CPF3C90 E</td>
<td>Literal value cannot be changed.</td>
</tr>
<tr>
<td>CPF3CF1 E</td>
<td>Error code parameter not valid.</td>
</tr>
<tr>
<td>CPF8100 E</td>
<td>All CPF81xx messages could be returned. xx is from 01 to FF.</td>
</tr>
<tr>
<td>CPF9681 E</td>
<td>Activity &amp;1 already exists.</td>
</tr>
<tr>
<td>CPF9682 E</td>
<td>Element &amp;3 of destination array not valid.</td>
</tr>
<tr>
<td>CPF9683 E</td>
<td>Number of conditions &amp;1 not valid.</td>
</tr>
<tr>
<td>CPF9684 E</td>
<td>Start after time &amp;1 not valid.</td>
</tr>
<tr>
<td>CPF9685 E</td>
<td>Start before time &amp;1 not valid.</td>
</tr>
<tr>
<td>CPF9686 E</td>
<td>Destination format value &amp;1 not valid.</td>
</tr>
<tr>
<td>CPF9687 E</td>
<td>Number of destinations &amp;1 not valid.</td>
</tr>
<tr>
<td>CPF9688 E</td>
<td>Element &amp;3 of condition list array not valid.</td>
</tr>
<tr>
<td>CPF9689 E</td>
<td>Hold value &amp;1 not valid.</td>
</tr>
<tr>
<td>CPF968A E</td>
<td>Activity name &amp;1 not valid.</td>
</tr>
<tr>
<td>CPF968B E</td>
<td>Activity type &amp;1 not valid.</td>
</tr>
</tbody>
</table>
Message ID  Error Message Text
CPF968C E  Replace value &1 not valid.
CPF968D E  Function parameters length &1 not valid.
CPF968E E  Condition list or start time cannot be specified.
CPF9691 E  Start after date &1 not valid.
CPF9692 E  Start before date &1 not valid.
CPF9696 E  Generated activity name limit exceeded.
CPF9697 E  Activity cannot be conditioned on itself.
CPF9698 E  Maximum size of CRQD &1 exceeded.
CPF9699 E  Start time not valid.
CPF969E E  Internal processing error occurred.
CPF969F E  Activity &1 already exists in condition list.
CPF96A2 E  CRQD library name &1 not valid.
CPF96A4 E  Activity not added, limit exceeded.
CPF9801 E  Object &2 in library &3 not found.
CPF9802 E  Not authorized to object &2 in &3.
CPF9803 E  Cannot allocate object &2 in library &3.
CPF9804 E  Object &2 in library &3 damaged.
CPF9807 E  One or more libraries in library list deleted.
CPF9808 E  Cannot allocate one or more libraries on library list.
CPF9810 E  Library &1 not found.
CPF9820 E  Not authorized to use library &1.
CPF9830 E  Cannot assign library &1.
CPF9838 E  User profile storage limit exceeded.
CPF9872 E  Program or service program &1 in library &2 ended. Reason code &3.

API introduced: V3R1

List Activities (QFVLSTA) API

Required Parameter Group:

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

Default Public Authority: *USE
Threadsafe: No

The List Activities (QFVLSTA) API retrieves a list of activities from a qualified change request description.

Authorities and Locks

Change request description authority
*USE

Change request description lock
*SHRRD

User space authority
*CHANGE

User space library authority
*EXECUTE
Required Parameter Group

Qualified user space name
   INPUT; CHAR(20)
   The name of the qualified user space that receives the generated list. The first 10 characters contain the user space name, the second 10 characters contain the user space library name.
   Special values for the user space library name are:
   *
   CURLIB The current library
   *LIBL The library list

Format
   INPUT; CHAR(8)
   The format of the data to return. The valid values are:
   CRDA0100 The activity name is returned. See “CRDA0100 Format” on page 63.
   CRDA0200 The activity name, start time, and conditions are returned. See “CRDA0200 Format” on page 63.
   CRDA0300 The activity name, type, text, conditions, and function parameters are returned. See “CRDA0300 Format” on page 63.

Qualified change request description name
   INPUT; CHAR(20)
   The name of the qualified change request description from which information will be retrieved. The first 10 characters contain the change request description name and the second 10 characters contain the change request description library name.
   Special values for the qualified change request description library name are:
   *
   CURLIB The current library
   *LIBL The library list

Error code
   I/O; CHAR(*)
   The structure in which to return error information. For the format of the structure, see Error Code Parameter.

Format of the Generated List
The user space is used to retrieve the activity list with the requested information and consists of:
   • A user area
   • A generic header
   • An input parameter section
   • A header section

For details about the user area and generic header, see User Space Format for List APIs. For details about the other items, see the following sections. For a detailed description of each field in the information returned, see “Field Descriptions” on page 64.
### Input Parameter Section

<table>
<thead>
<tr>
<th>Offset</th>
<th>Dec</th>
<th>Hex</th>
<th>Type</th>
<th>Field</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>CHAR(10)</td>
<td>User space name specified</td>
</tr>
<tr>
<td>10</td>
<td>A</td>
<td>1</td>
<td>CHAR(10)</td>
<td>User space library name specified</td>
</tr>
<tr>
<td>20</td>
<td>14</td>
<td>14</td>
<td>CHAR(8)</td>
<td>Format name specified</td>
</tr>
<tr>
<td>28</td>
<td>1C</td>
<td>20</td>
<td>CHAR(10)</td>
<td>Change request description name specified</td>
</tr>
<tr>
<td>38</td>
<td>26</td>
<td>30</td>
<td>CHAR(10)</td>
<td>Change request description library name specified</td>
</tr>
</tbody>
</table>

### Header Section

<table>
<thead>
<tr>
<th>Offset</th>
<th>Dec</th>
<th>Hex</th>
<th>Type</th>
<th>Field</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>CHAR(10)</td>
<td>User space name used</td>
</tr>
<tr>
<td>10</td>
<td>A</td>
<td>1</td>
<td>CHAR(10)</td>
<td>User space library name used</td>
</tr>
<tr>
<td>20</td>
<td>14</td>
<td>14</td>
<td>CHAR(10)</td>
<td>Change request description name used</td>
</tr>
<tr>
<td>30</td>
<td>1E</td>
<td>30</td>
<td>CHAR(10)</td>
<td>Change request description library name used</td>
</tr>
</tbody>
</table>

### CRDA0100 Format

See “Field Descriptions” on page 64 for descriptions of the fields in this format.

<table>
<thead>
<tr>
<th>Offset</th>
<th>Dec</th>
<th>Hex</th>
<th>Type</th>
<th>Field</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>CHAR(10)</td>
<td>Activity name</td>
</tr>
</tbody>
</table>

### CRDA0200 Format

See “Field Descriptions” on page 64 for descriptions of the fields in this format.

<table>
<thead>
<tr>
<th>Offset</th>
<th>Dec</th>
<th>Hex</th>
<th>Type</th>
<th>Field</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>CHAR(10)</td>
<td>Activity name</td>
</tr>
<tr>
<td>10</td>
<td>A</td>
<td>AA</td>
<td>Array of CHAR(32)</td>
<td>Conditions array</td>
</tr>
<tr>
<td>170</td>
<td>AA</td>
<td>170</td>
<td>CHAR(2)</td>
<td>Reserved</td>
</tr>
<tr>
<td>172</td>
<td>AC</td>
<td>172</td>
<td>BINARY(4)</td>
<td>Number of conditions</td>
</tr>
<tr>
<td>176</td>
<td>BO</td>
<td>176</td>
<td>CHAR(40)</td>
<td>Start time</td>
</tr>
</tbody>
</table>

### CRDA0300 Format

See “Field Descriptions” on page 64 for descriptions of the fields in this format.
### Field Descriptions

**Activity name.** The name of the activity stored in the change request description name.

**Activity text description.** The description of the activity.

**Activity type.** The name of the activity type stored in the change request description name. This value is defined by the user program. The API does not define the possible values.

**Change request description name specified.** The name of the change request description specified to the API.

**Change request description name used.** The name of the change request description used to retrieve the list.

**Change request description library name specified.** The name of the library where the change request description is stored specified to the API. If the special values *LIBL or *CURLIB are used, the library name resolved is set here.

**Change request description library name used.** The name of the library used where the change request description is stored.

**Conditions array.** The list of conditions that must be met before the activity can be run. This is an array of 5 elements. The number of conditions field indicates the actual number of conditions specified for the activity.

The structure of each element is:

- **activity** CHAR(10)
  - Activity name. Possible values: *PRV, user-specified name, or a generic name (partial name followed by an asterisk (*)).

- **relation** CHAR(3)

- **reserved** An ignored field.
code

BIN(31)

The end code value to base conditioning on. The following values are supported:

0-99  User can specify anything in this range.

-1 (*SUCCESS)
   The end code is any value from 0-9.
   If specified, relation must be *EQ or *NE.

-2 (*FAIL)
   The end code is any value from 10-89.
   If specified, relation must be *EQ or *NE.

-3 (*NOTRUN)
   The end code is any value from 90-99.
   If specified, relation must *EQ or *NE.

-4 (*ANY)
   The end code is any value from 0-99.
   If specified, relation must be *EQ.

mode

CHAR(10)

Condition mode. Possible values are:

*ALLNODES
   All nodes of the conditioning activity must meet the completion criteria before this
   condition is considered met.

*SAMENODE
   Nodes of the conditioned activity need only wait for the same node of the conditioning
   activity to meet the completion criteria before this condition is considered met.

CCSID of activity text. The CCSID of the activity text description.

Format name specified. The format name specified to the API.

Function parameters. The parameters defined by the calling application when the activity was added.

Function parameters length. The length of the function parameters returned.

Number of conditions. The number of conditions specified for the activity. Valid values range from 0
through 5. This field is used to determine how many elements of the condition array contain valid data.

Reserved. An ignored field.

Start time. The date and time window during which this activity can be started. The current date and
time values and next date values are determined when change request is submitted. The structure is:

start after time  CHAR(10)

The time after which this activity may be started. Special value supported is:

*CURRENT
   This activity may start any time on or after the time at which the change request was
   submitted.
**start after date**  
CHAR(10)

The date after which this activity may be started. Special values supported are:

*CURRENT  
This activity may start on any date on or after the date on which the change request was submitted.

*NEXT   
This activity may start on any date after the date on which the change request was submitted.

**start before time**  
CHAR(10)

The time before which the activity must be started. If the activity cannot be started before this time then it will never be started. Special values supported are:

*ANY  
The activity may start any time on or after the start time.

*CURRENT  
The activity must start before the time the change request was submitted on the date specified on the start before date field.

**start before date**  
CHAR(10)

The date before which the activity must be started. If the activity cannot be started by this date then it will never be started. Special values supported are:

*ANY  
The activity may start at any date on or after the start date.

*CURRENT  
The activity must start on the date the change request was submitted.

*NEXT  
The activity must start by the day after the date the change request was submitted.

The format of time is HHMMSS and the format of date is CYYMMDD.

**User space library name specified.** The user space library name specified to the API.

**User space library name used.** The names of the actual user space library used to report data. If the special values *LIBL or *CURLIB are used, the library name resolved is set here.

**User space name specified.** The user space name specified to the API.

**User space name used.** The actual user space name used to report data.

### Error Messages

<table>
<thead>
<tr>
<th>Message ID</th>
<th>Error Message Text</th>
</tr>
</thead>
<tbody>
<tr>
<td>CPF24B4 E</td>
<td>Severe error while addressing parameter list.</td>
</tr>
<tr>
<td>CPF3793 E</td>
<td>Machine storage limit reached.</td>
</tr>
<tr>
<td>CPF3C21 E</td>
<td>Format name &amp;1 is not valid.</td>
</tr>
<tr>
<td>CPF3C90 E</td>
<td>Literal value cannot be changed.</td>
</tr>
<tr>
<td>CPF3CAA E</td>
<td>List is too large for user space &amp;1.</td>
</tr>
<tr>
<td>CPF3CF1 E</td>
<td>Error code parameter not valid.</td>
</tr>
<tr>
<td>CPF8100 E</td>
<td>All CPF81xx messages could be returned. xx is from 01 to FF.</td>
</tr>
<tr>
<td>CPF969E E</td>
<td>Internal processing error occurred.</td>
</tr>
<tr>
<td>CPF96A1 E</td>
<td>User space library name &amp;1 not valid.</td>
</tr>
<tr>
<td>CPF96A2 E</td>
<td>CRQD library name &amp;1 not valid.</td>
</tr>
<tr>
<td>CPF9801 E</td>
<td>Object &amp;2 in library &amp;3 not found.</td>
</tr>
<tr>
<td>CPF9802 E</td>
<td>Not authorized to object &amp;2 in &amp;3.</td>
</tr>
<tr>
<td>CPF9803 E</td>
<td>Cannot allocate object &amp;2 in library &amp;3.</td>
</tr>
<tr>
<td>CPF9804 E</td>
<td>Object &amp;2 in library &amp;3 damaged.</td>
</tr>
</tbody>
</table>
Remove Activity (QFVRMVA) API

Required Parameter Group:

1  Qualified change request description name  Input  Char(20)
2  Activity name  Input  Char(10)
3  Error code  I/O  Char(*)

Default Public Authority: *USE
Threadsafe: No

The Remove Activity (QFVRMVA) API removes an activity from the specified change request description.

Authorities and Locks

Change request description authority
*CHANGE

Change request description lock
*EXCLRD

Library Authority
*EXECUTE

Required Parameter Group

Qualified change request description name
INPUT;CHAR(20)

The name of the change request description from which an activity is removed. The first 10 characters contain the name of the change request description and the second 10 characters contain the change request description library name.

Special values for the change request description library name are:

*CURLIB  The current library
*LIBL    The library list
Activity name
   INPUT; CHAR(10)

   The name of the activity to remove from the change request description.

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

<table>
<thead>
<tr>
<th>Message ID</th>
<th>Error Message Text</th>
</tr>
</thead>
<tbody>
<tr>
<td>CPF2150 E</td>
<td>Object information function failed.</td>
</tr>
<tr>
<td>CPF2151 E</td>
<td>Operation failed for &amp;2 in &amp;1 type *&amp;3.</td>
</tr>
<tr>
<td>CPF24B4 E</td>
<td>Severe error while addressing parameter list.</td>
</tr>
<tr>
<td>CPF3C90 E</td>
<td>Literal value cannot be changed.</td>
</tr>
<tr>
<td>CPF3CF1 E</td>
<td>Error code parameter not valid.</td>
</tr>
<tr>
<td>CPF8100 E</td>
<td>All CPF81xx messages could be returned. xx is from 01 to FF.</td>
</tr>
<tr>
<td>CPF9693 E</td>
<td>Activity &amp;1 not found.</td>
</tr>
<tr>
<td>CPF96A2 E</td>
<td>CRQD library name &amp;1 not valid.</td>
</tr>
<tr>
<td>CPF9801 E</td>
<td>Object &amp;2 in library &amp;3 not found.</td>
</tr>
<tr>
<td>CPF9802 E</td>
<td>Not authorized to object &amp;2 in &amp;3.</td>
</tr>
<tr>
<td>CPF9803 E</td>
<td>Cannot allocate object &amp;2 in library &amp;3.</td>
</tr>
<tr>
<td>CPF9804 E</td>
<td>Object &amp;2 in library &amp;3 damaged.</td>
</tr>
<tr>
<td>CPF9807 E</td>
<td>One or more libraries in library list deleted.</td>
</tr>
<tr>
<td>CPF9808 E</td>
<td>Cannot allocate one or more libraries on library list.</td>
</tr>
<tr>
<td>CPF9810 E</td>
<td>Library &amp;1 not found.</td>
</tr>
<tr>
<td>CPF9820 E</td>
<td>Not authorized to use library &amp;1.</td>
</tr>
<tr>
<td>CPF9830 E</td>
<td>Cannot assign library &amp;1.</td>
</tr>
<tr>
<td>CPF9872 E</td>
<td>Program or service program &amp;1 in library &amp;2 ended. Reason code &amp;3.</td>
</tr>
</tbody>
</table>

API introduced: V3R1

Retrieve Change Request Description (QFVRTVCD) API

Required Parameter Group:

- **1** Receiver variable       Output       Char(*)
- **2** Length of receiver variable  Input       Binary(4)
- **3** Format                   Input       Char(8)
- **4** Qualified change request description name  Input       Char(20)
- **5** Activity name            Input       Char(10)
- **6** Error code               I/O       Char(*)

Default Public Authority: *USE
Threadsafe: No

The Retrieve Change Request Description (QFVRTVCD) API retrieves either general change request description information, or information pertaining to a single activity within the change request description.
Authorities and Locks

Change request description authority
*USE

Change request description lock
*SHRRD

Required Parameter Group

Receiver variable
OUTPUT; CHAR(*)

The receiver variable that receives the information requested. You can specify the size of the area to be smaller than the format requested as long as you specify the length parameter correctly. As a result, the API returns only the data that the area can hold. See Table 1 and Table 2 for descriptions of the formats.

Length of receiver variable
INPUT; BINARY(4)

The length of the receiver variable provided. The length of receiver variable parameter may be specified up to the size of the receiver variable specified in the user program. If the length of receiver variable parameter specified is larger than the allocated size of the receiver variable specified in the user program, the results are not predictable. The minimum length is 8 bytes.

Format
INPUT; CHAR(8)

The format of the data to return. The possible formats are:

CRQD0100 Returns general change request description information.
CRQD0200 Returns specific activity information for the specified change request description.

Qualified change request description name
INPUT; CHAR(20)

The qualified name of the change request description to retrieve information for. The Format is:

name CHAR(10)
library CHAR(10)

Possible values: *LIBL, *CURLIB, library name.

Activity name
INPUT; CHAR(10)

The name of the activity to retrieve information for. This is ignored when the format parameter is CRQD0100.

Error code
I/O; CHAR(*)

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

CRQD0100 Format
See “Field Descriptions” on page 71 for descriptions of the fields in this format.
<table>
<thead>
<tr>
<th>Offset</th>
<th>Dec</th>
<th>Hex</th>
<th>Type</th>
<th>Field</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>BIN(4)</td>
<td>Bytes returned</td>
</tr>
<tr>
<td>4</td>
<td>4</td>
<td>4</td>
<td>BIN(4)</td>
<td>Bytes available</td>
</tr>
<tr>
<td>8</td>
<td>8</td>
<td>8</td>
<td>CHAR(10)</td>
<td>Qualified change request description name</td>
</tr>
<tr>
<td>18</td>
<td>12</td>
<td>12</td>
<td>CHAR(10)</td>
<td>Qualified change request description library name</td>
</tr>
<tr>
<td>28</td>
<td>1C</td>
<td>26</td>
<td>CHAR(10)</td>
<td>Owner’s profile</td>
</tr>
<tr>
<td>38</td>
<td>30</td>
<td>30</td>
<td>CHAR(10)</td>
<td>User profile to run under</td>
</tr>
<tr>
<td>54</td>
<td>36</td>
<td>36</td>
<td>CHAR(10)</td>
<td>Problem ID</td>
</tr>
<tr>
<td>64</td>
<td>40</td>
<td>40</td>
<td>CHAR(21)</td>
<td>Problem origin</td>
</tr>
<tr>
<td>85</td>
<td>55</td>
<td>55</td>
<td>CHAR(50)</td>
<td>Text description</td>
</tr>
</tbody>
</table>

**CRQD0200 Format**

See “Field Descriptions” on page 71 for descriptions of the fields in this format.
### Field Descriptions

**Activity name.** The name of the activity for which information is retrieved.

**Activity text.** The text description of the activity specified.

**Activity type.** The name of the activity type for this activity.

**Bytes available.** The number of bytes of data available to the calling program. The receiver variable could get this much data from this format if the receiver variable was this large or larger.

**Bytes returned.** The number of bytes of data that were returned to the calling program in the receiver variable. If this value is smaller than bytes available, there was more data than there was room for.

**CCSID of activity text description.** The coded character set identifies (CCSID) used for the activity text description.

**Conditions list.** The list of conditions that must be met before the activity can run. This is an array of up to 5 elements. The number of elements is specified by the number of conditions field. There can be up to 5 elements. If this data is not present, number of conditions and offset to conditions is 0.

Each element has the following format:

- **activity**
  - **Activity name**
  - **relation**
  - **reserved code**

<table>
<thead>
<tr>
<th>activity</th>
<th>CHAR(10)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Activity name</td>
<td>Possible values are: *PRV, user specified name.</td>
</tr>
<tr>
<td>relation</td>
<td>CHAR(3)</td>
</tr>
<tr>
<td>reserved code</td>
<td>BIN(4)</td>
</tr>
<tr>
<td>The relational operator. Possible values are:</td>
<td></td>
</tr>
<tr>
<td>The end code of conditioning activity. Valid values are 0-99. The following special values are also supported:</td>
<td></td>
</tr>
<tr>
<td>-1(*SUCCESS)</td>
<td></td>
</tr>
<tr>
<td>The end code is any value from 0-9</td>
<td></td>
</tr>
<tr>
<td>-2(*FAIL)</td>
<td></td>
</tr>
<tr>
<td>The end code is any value from 10-89</td>
<td></td>
</tr>
<tr>
<td>-3(*NOTRUN)</td>
<td></td>
</tr>
<tr>
<td>The end code is any value from 90-99.</td>
<td></td>
</tr>
<tr>
<td>-4 (*ANY)</td>
<td></td>
</tr>
<tr>
<td>The end code is any value from 0-99.</td>
<td></td>
</tr>
</tbody>
</table>

---

**Note:** The offset varies. This field is either optional or variable length. The offset is specified in a previous field. The data is left-justified with a blank pad at the end. The array of destination nodes and the array of conditions are returned in the same sequence they were entered.
Condition mode. Valid values are:

*ALLNODES
   All nodes of the conditioning activity must meet the completion criteria before this condition is considered met.

*SAMENODE
   Nodes of the conditioned activity need only wait for the same node of the conditioning activity to meet the completion criteria before this condition is considered met.

Reserved for boundary alignment.

Destination is *LCL. Whether the destination for this activity is the local system. Possible values are Y and N.

Destination nodes. An array of elements. The number of elements is specified by the number of nodes field. There can be up to 50 elements. If number of nodes is 0, this field will not exist.

Each element has the following format:

   Network ID   CHAR(8)
   Control point CHAR(8)
   Reserved     CHAR(4)

Function parameters. The structure is defined by the calling program.

Hold. Whether the activity is held when the change request is submitted. Possible values are:

*YES   The activity is held when the change request is submitted. The user must release the activity before it will run.
*NO    The activity is not held when the change request is submitted.

Length of function parameters. The length of the function parameters.

Node list name. The name of the node list. If no node list name exists, the offset to node list name is 0.

Node list library name. The library name of the node list.

Number of conditions. The number of conditions that must be satisfied before the activity can run. Possible values are 0 to 5. If no conditions are present, number of conditions and the offset to conditions are 0.

Number of nodes. The number of nodes in the destination nodes field. The possible values are 1 to 50.

Offset to conditions. If no conditions are present, number of conditions and the offset to conditions are 0.

Offset to destination nodes. The offset to the list of destination nodes.

Offset to function parameters. The offset to the function parameters.

Offset to node list name. Offset to the node list name. If this is 0, the node list name is not present.

Owner’s profile. User profile that created the change request description object.
**Problem ID.** The ID of the problem associated with the change request description retrieved.

**Problem origin.** The origin system of the problem ID. The Format is:

- **Origin type**: CHAR(1)
  
  Currently, the only value supported is A, which indicates the transport type is APPN.

- **Network ID**: CHAR(8)

- **Control point**: CHAR(8)

- **Reserved**: CHAR(4)

**Qualified change request description name.** The qualified name of the change request description object for which data is retrieved. The Format is:

- **Name**: CHAR(10)
- **Library**: CHAR(10)

**Sequence number.** The sequence number that is used to generate a generic activity name at the time an activity is added to the change request description.

**Start time.** The date and time window during which this activity can be started. The current date and time values and next date values are determined when the change request is submitted. The format is:

- **Start after time**: CHAR(10)
  
  The time after which this activity may be started. Special values supported:
  
  * **CURRENT**
    
    This activity may start any time on or after the time at which the change request was submitted.

- **Start after date**: CHAR(10)
  
  The date after which this activity may be started. Special values supported:
  
  * **CURRENT**
    
    This activity may start any date on or after the date on which the change request was submitted.

  * **NEXT**
    
    This activity may start on any date after the date on which the change request was submitted.

- **Start before time**: CHAR(10)
  
  The time before which the activity must be started. If the activity cannot be started before this time then it is never started. Special values supported:
  
  * **ANY**
    
    The activity may start any time on or after the start time.

  * **CURRENT**
    
    The activity must start before the time the change request was submitted on the date specified on the start before date field.
**Start before date**  CHAR(10)

The date before which the activity must be started. If the activity cannot be started by this date then it is never started. Special values supported:

*ANY  The activity may start at any date on or after the start date.
*CURRENT The activity must start on the date the change request was submitted.
*NEXT  The activity must start by the day after the date the change request was submitted.

The format of time is HHMMSS, where:

- **HH** Hour. Value: 00-23
- **MM** Minute. Value: 00-59
- **SS** Second. Value: 00-59

The format of date is CYYMMDD, where:

- **C** Century, where 0 indicates years 19xx and 1 indicates years 20xx.
- **YY** Year. Value: 00-99
- **MM** Month. Value: 01-12
- **DD** Day. Value: 01-31

**Text description.** Text description of the change request description specified.

**User profile to run under.** Whether the authority checking done while this change request is running is based on the user who submitted the change request or the owner of the change request description. Possible values are:

- *OWNER  Change request runs under the user profile of the owner of the change request description.
- *SBM    Change request runs under the submitter of the change request.

**Error Messages**

The following messages are possible as either escape messages or return codes.

<table>
<thead>
<tr>
<th>Message ID</th>
<th>Error Message Text</th>
</tr>
</thead>
<tbody>
<tr>
<td>CPF2150 E</td>
<td>Object information function failed.</td>
</tr>
<tr>
<td>CPF2151 E</td>
<td>Operation failed for &amp;2 in &amp;1 type *&amp;3.</td>
</tr>
<tr>
<td>CPF24B4 E</td>
<td>Severe error while addressing parameter list.</td>
</tr>
<tr>
<td>CPF3C19 E</td>
<td>Error occurred with receiver variable specified.</td>
</tr>
<tr>
<td>CPF3C21 E</td>
<td>Format name &amp;1 is not valid.</td>
</tr>
<tr>
<td>CPF3C24 E</td>
<td>Length of the receiver variable is not valid.</td>
</tr>
<tr>
<td>CPF3C90 E</td>
<td>Literal value cannot be changed.</td>
</tr>
<tr>
<td>CPF3CF1 E</td>
<td>Error code parameter not valid.</td>
</tr>
<tr>
<td>CPF8100 E</td>
<td>All CPF81xx messages could be returned. xx is from 01 to FF.</td>
</tr>
<tr>
<td>CPF9693 E</td>
<td>Activity &amp;1 not found.</td>
</tr>
<tr>
<td>CPF9694 E</td>
<td>Activity name required.</td>
</tr>
<tr>
<td>CPF969E E</td>
<td>Internal processing error occurred.</td>
</tr>
<tr>
<td>CPF96A2 E</td>
<td>CRQD library name &amp;1 not valid.</td>
</tr>
<tr>
<td>CPF9801 E</td>
<td>Object &amp;2 in library &amp;3 not found.</td>
</tr>
<tr>
<td>CPF9802 E</td>
<td>Not authorized to object &amp;2 in &amp;3.</td>
</tr>
<tr>
<td>CPF9803 E</td>
<td>Cannot allocate object &amp;2 in library &amp;3.</td>
</tr>
<tr>
<td>Message ID</td>
<td>Error Message Text</td>
</tr>
<tr>
<td>------------</td>
<td>--------------------</td>
</tr>
<tr>
<td>CPF9804 E</td>
<td>Object &amp;2 in library &amp;3 damaged.</td>
</tr>
<tr>
<td>CPF9807 E</td>
<td>One or more libraries in library list deleted.</td>
</tr>
<tr>
<td>CPF9808 E</td>
<td>Cannot allocate one or more libraries on library list.</td>
</tr>
<tr>
<td>CPF9810 E</td>
<td>Library &amp;1 not found.</td>
</tr>
<tr>
<td>CPF9820 E</td>
<td>Not authorized to use library &amp;1.</td>
</tr>
<tr>
<td>CPF9830 E</td>
<td>Cannot assign library &amp;1.</td>
</tr>
<tr>
<td>CPF9872 E</td>
<td>Program or service program &amp;1 in library &amp;2 ended. Reason code &amp;3.</td>
</tr>
</tbody>
</table>

API introduced: V3R1
Appendix. Notices

This information was developed for products and services offered in the U.S.A.

IBM may not offer the products, services, or features discussed in this document in other countries. Consult your local IBM representative for information on the products and services currently available in your area. Any reference to an IBM product, program, or service is not intended to state or imply that only that IBM product, program, or service may be used. Any functionally equivalent product, program, or service that does not infringe any IBM intellectual property right may be used instead. However, it is the user’s responsibility to evaluate and verify the operation of any non-IBM product, program, or service.

IBM may have patents or pending patent applications covering subject matter described in this document. The furnishing of this document does not grant you any license to these patents. You can send license inquiries, in writing, to:

IBM Director of Licensing
IBM Corporation
North Castle Drive
Armonk, NY 10504-1785
U.S.A.

For license inquiries regarding double-byte (DBCS) information, contact the IBM Intellectual Property Department in your country or send inquiries, in writing, to:

IBM World Trade Asia Corporation
Licensing
2-31 Roppongi 3-chome, Minato-ku
Tokyo 106-0032, Japan

The following paragraph does not apply to the United Kingdom or any other country where such provisions are inconsistent with local law: INTERNATIONAL BUSINESS MACHINES CORPORATION PROVIDES THIS PUBLICATION “AS IS” WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESS OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF NON-INFRINGEMENT, MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE. Some states do not allow disclaimer of express or implied warranties in certain transactions, therefore, this statement may not apply to you.

This information could include technical inaccuracies or typographical errors. Changes are periodically made to the information herein; these changes will be incorporated in new editions of the publication. IBM may make improvements and/or changes in the product(s) and/or the program(s) described in this publication at any time without notice.

Any references in this information to non-IBM Web sites are provided for convenience only and do not in any manner serve as an endorsement of those Web sites. The materials at those Web sites are not part of the materials for this IBM product and use of those Web sites is at your own risk.

IBM may use or distribute any of the information you supply in any way it believes appropriate without incurring any obligation to you.

Licensees of this program who wish to have information about it for the purpose of enabling: (i) the exchange of information between independently created programs and other programs (including this one) and (ii) the mutual use of the information which has been exchanged, should contact:
Such information may be available, subject to appropriate terms and conditions, including in some cases, payment of a fee.

The licensed program described in this information and all licensed material available for it are provided by IBM under terms of the IBM Customer Agreement, IBM International Program License Agreement, IBM License Agreement for Machine Code, or any equivalent agreement between us.

Any performance data contained herein was determined in a controlled environment. Therefore, the results obtained in other operating environments may vary significantly. Some measurements may have been made on development-level systems and there is no guarantee that these measurements will be the same on generally available systems. Furthermore, some measurements may have been estimated through extrapolation. Actual results may vary. Users of this document should verify the applicable data for their specific environment.

Information concerning non-IBM products was obtained from the suppliers of those products, their published announcements or other publicly available sources. IBM has not tested those products and cannot confirm the accuracy of performance, compatibility or any other claims related to non-IBM products. Questions on the capabilities of non-IBM products should be addressed to the suppliers of those products.

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

COPYRIGHT LICENSE:

This information contains sample application programs in source language, which illustrate programming techniques on various operating platforms. You may copy, modify, and distribute these sample programs in any form without payment to IBM, for the purposes of developing, using, marketing or distributing application programs conforming to the application programming interface for the operating platform for which the sample programs are written. These examples have not been thoroughly tested under all conditions. IBM, therefore, cannot guarantee or imply reliability, serviceability, or function of these programs.

Each copy or any portion of these sample programs or any derivative work, must include a copyright notice as follows:

(C) IBM 2006. Portions of this code are derived from IBM Corp. Sample Programs. (C) Copyright IBM Corp. 1998, 2006. All rights reserved.

If you are viewing this information softcopy, the photographs and color illustrations may not appear.

**Programming Interface Information**

This Application Programming Interfaces (API) publication documents intended Programming Interfaces that allow the customer to write programs to obtain the services of IBM i5/OS.
Trademarks

The following terms are trademarks of International Business Machines Corporation in the United States, other countries, or both:
Advanced 36
Advanced Function Printing
Advanced Peer-to-Peer Networking
AFP
AIX
AS/400
COBOL/400
CUA
DB2
DB2 Universal Database
Distributed Relational Database Architecture
Domino
DPI
DRDA
eServer
GDDM
IBM
Integrated Language Environment
Intelligent Printer Data Stream
IPDS
i5/OS
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**

Permissions for the use of these Publications is granted subject to the following terms and conditions.

Personal Use: You may reproduce these Publications for your personal, noncommercial use provided that all proprietary notices are preserved. You may not distribute, display or make derivative works of these Publications, or any portion thereof, without the express consent of IBM.

Commercial Use: You may reproduce, distribute and display these Publications solely within your enterprise provided that all proprietary notices are preserved. You may not make derivative works of these Publications, or reproduce, distribute or display these Publications or any portion thereof outside your enterprise, without the express consent of IBM.

Except as expressly granted in this permission, no other permissions, licenses or rights are granted, either express or implied, to the Publications or any information, data, software or other intellectual property contained therein.

IBM reserves the right to withdraw the permissions granted herein whenever, in its discretion, the use of the Publications is detrimental to its interest or, as determined by IBM, the above instructions are not being properly followed.

You may not download, export or re-export this information except in full compliance with all applicable laws and regulations, including all United States export laws and regulations. IBM MAKES NO GUARANTEE ABOUT THE CONTENT OF THESE PUBLICATIONS. THE PUBLICATIONS ARE PROVIDED "AS-IS" AND WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESSED OR IMPLIED, INCLUDING BUT NOT LIMITED TO IMPLIED WARRANTIES OF MERCHANTABILITY, NON-INFRINGEMENT, AND FITNESS FOR A PARTICULAR PURPOSE.