

IBM i
7.2

Networking
Open Shortest Path First (OSPF) support



Note

Before using this information and the product it supports, read the information in [“Notices” on page 25.](#)

This document may contain references to Licensed Internal Code. Licensed Internal Code is Machine Code and is licensed to you under the terms of the IBM License Agreement for Machine Code.

© **Copyright International Business Machines Corporation 1998, 2013.**

US Government Users Restricted Rights – Use, duplication or disclosure restricted by GSA ADP Schedule Contract with IBM Corp.

Contents

Open Shortest Path First support.....	1
What's new for IBM i 7.1.....	1
PDF file for Open Shortest Path First support.....	1
Open Shortest Path First support concepts.....	2
OSPF routing domain and areas.....	2
OSPF area aggregation.....	4
Link-state advertisements.....	5
Aging of link-state records.....	6
Packet types for OSPF.....	7
OSPF for IPv6.....	8
OSPF interfaces.....	9
Point-to-point links for OSPF.....	9
i5/OS OSPF Authentication.....	10
Enabling of i5/OS OSPF job tracing.....	11
Open Shortest Path First support tasks.....	12
Configuring i5/OS for OSPF networking.....	12
Enabling TCP/IP for OSPF on i5/OS.....	12
Open Shortest Path First support reference.....	13
Open Shortest Path First API and commands.....	13
Scenarios: Configuring OSPF.....	15
Scenario: Configuring OSPF interfaces and neighbors.....	15
Scenario: OSPF multipath routes.....	18
Scenario: Retrieve OSPF State Information API.....	20
Notices.....	25
Programming interface information.....	26
Trademarks.....	26
Terms and conditions.....	27

Open Shortest Path First support

i5/OS support includes the Open Shortest Path First (OSPF) protocol. OSPF is a link-state, hierarchical Interior Gateway Protocol (IGP) for network routing.

This topic collection describes i5/OS support for OSPF configuration, authentication methods, point-to-point links, packet types and splitting an OSPF autonomous system (AS) into areas. It also includes three scenarios, one that demonstrates OSPF routes on a TCP/IP stack, one that demonstrates multipath routes, and another that demonstrates an i5/OS API.

Note: By using the code examples, you agree to the terms of the [“Code license and disclaimer information”](#) on page 24.

Related information

[Open Shortest Path First](#)



What's new for IBM i 7.1

Read about new or significantly changed information for the Open Shortest Path First (OSPF) topic collection.

Miscellaneous technical changes have been made since the previous publication.

How to see what's new or changed

To help you see where technical changes have been made, the information center uses:

- The  image to mark where new or changed information begins.
- The  image to mark where new or changed information ends.

In PDF files, you might see revision bars (|) in the left margin of new and changed information.

To find other information about what's new or changed this release, see the [Memo to users](#).

PDF file for Open Shortest Path First support

You can view and print a PDF file of this information.


To view or download the PDF version of this document, select [Open Shortest Path First \(OSPF\) support](#) (about 205 KB).

Saving PDF files

To save a PDF on your workstation for viewing or printing:

1. Right-click the PDF link in your browser.
2. Click the option that saves the PDF locally.
3. Navigate to the directory in which you want to save the PDF.
4. Click **Save**.

Downloading Adobe Reader

You need Adobe Reader installed on your system to view or print these PDFs. You can download a free copy from the [Adobe Web site](http://www.adobe.com/products/acrobat/readstep.html) (www.adobe.com/products/acrobat/readstep.html) .

Open Shortest Path First support concepts

Before using i5/OS support in an OSPF networking environment, it is important to understand the requirements of an OSPF participant. This topic describes the OSPF concept of areas and interfaces, and identifies the i5/OS commands that configure these on your system.

OSPF uses a link-state algorithm to calculate and use the shortest distance between a central node and all other nodes in an OSPF networking environment. Each node describes and sends the state of its own links and the complete topology, or routing structure, created by it and all of its neighbor nodes.

The i5/OS OSPF support uses the OMPROUTED server to register the system as a node or participant in an OSPF network. Each OSPF node or route destination is in the format of a dotted decimal IPv4 or IPv6 address. The OMPROUTED server daemon conforms to UNIX standards and is POSIX compliant. It uses the TCP/IP protocols provided on the i5/OS platform to run the QTOORROUTE job in the QSYSWRK subsystem. The job supports both IPv4 and IPv6 operations.

OSPF concepts include areas and interfaces to routers that are defined within them. The OSPF protocol uses packet types for initial neighbor discovery, and to send and receive link-state advertisements (LSAs) throughout its network of IP addresses. It uses different kinds of routers and path types to support different types of networks. Before using i5/OS OSPF support, determine how to split an OSPF routing domain into areas and how to use other OSPF functions, such as area aggregation and LSAs.

The i5/OS support is different for IPv4 and IPv6, when using OSPF routing. This topic covers these differences and other important information like the use of an i5/OS line description to provide a TCP/IP OSPF interface.

OSPF routing domain and areas

OSPF routing depends on the relationship that is defined between areas within a routing domain. CL commands are used to define area types and i5/OS routing neighbors.

For the OSPF protocol, an *autonomous system (AS)* is a collection of IP networks that are under a common administration, sharing a common routing strategy, and running only one routing protocol. The routers and links that make up the AS are in logical groups called areas. Areas are identified by uniquely assigned numbers and an AS must define at least one area.

When an AS is divided into multiple areas, the areas are interconnected by a router that is designated as an *area border router (ABR)*. By definition, an ABR has different OSPF interfaces that are attached to different OSPF areas so that it can operate in more than one area. The ABR keeps a copy of the link-state database for each associated area.

All routers in an area have identical copies of the autonomous system topology database. Each router computes its own routing table using a spanning tree algorithm, called the Dijkstra algorithm, that computes the Shortest Path First.

OSPF supports multipath, which means it supports multiple routes to the same destination or system. When a new link-state update packet is received, the entire tree is recomputed. Other multipath calculation considerations include the following:

- The routing tree retains all multiple equal-cost routes.
- The routing tree selects only the shortest path for retention when multiple routes exists to the same destination.
- Both costs are added to the stack when two routes, that have the same costs, are to the same destination.
- OSPF always adds the route with the lowest cost to the TCP/IP stack.
- Multipath routes to the same destination are added to the TCP/IP routing table when those paths have the same cost.

Each router in an OSPF area originates a *link-state advertisement (LSA)*, which is a basic means of OSPF communication, to transport the topology of one router to all other routers in the same OSPF area. Each link-state originated advertisement, in each router, is stored in its link-state database. The database is

synchronized between routers so that each router of the OSPF area, has an identical copy of the link-state database.

An ABR uses a separate, summary-LSA to advertise to all other destinations that are known to the ABR but that are outside of the area.

An area designated as the *backbone area* is a requirement in an OSPF AS. All areas must connect to the backbone because it is responsible for providing routing information between all areas in the OSPF AS. This backbone or tree-like design supports routing from any ABR to any other ABR by traversing only backbone network segments.

Areas are identified by an ID that is four octets in length. The backbone area is always assigned to the reserved ID of 0.0.0.0. All other areas in an AS must have a unique identifier. By definition an OSPF area is a collection of networks, not a collection of routers. A backbone network segment is an IP subnet that belongs to the area identified by 0.0.0.0. Areas that are not physically connected to the backbone are logically connected by a backbone ABR using an OSPF virtual link. The Add OSPF Area (ADDOSPFARA) command adds i5/OS backbone and nonbackbone areas to an OSPF configuration.

Depending on the size of the routing trees, OSPF is compute-intensive so it is common to populate only nonbackbone areas with application servers. The topology of one area is hidden from the rest of the areas of an AS to reduce routing overhead, because fewer routing updates are sent and smaller routing trees are computed and maintained. This also leads to a reduction in storage and CPU consumption.

Figure 1. Split an OSPF routing domain into areas.

In the example, the backbone area is located at IP address 9.7.85.0. Three nonbackbone areas, area 1.1.1.1, area 2.2.2.2, and area 3.3.3.3, are connected, with area 3.3.3.3 connected to the backbone area through a virtual link.

OSPF backbone, nonbackbone, and stub areas

The backbone area requirement is that the distribution of routing information is done from subnet areas to the backbone area and then the backbone sends this information to the rest of the nonbackbone areas. To distribute routing information through different areas, the ABR does calculations from entries of routing tables, which are summary-LSAs, that are received from the backbone area. The ABR then sends this routing information, also in LSA-summaries, to its attached nonbackbone areas.

Virtual links are configured between ABRs and are considered point-to-point interfaces to connect nonbackbone areas to the OSPF backbone area 0.0.0.0, when there is not a physical connection between them. When the virtual link reaches full adjacency with a neighbor through a nonbackbone area, this nonbackbone area is advertised as a transit area. Transit areas carry traffic that is not originated in or destined for the area itself.

An area that is configured as a *stub area* does not support the flooding of external routing information by external LSAs. Stub areas are configured to omit external routing to reduce CPU cycles and to reduce the size of its link-state database. The only way an i5/OS stub area can reach external routing is by a default route. To configure a stub area, set the stub area option to *YES on the ADDOSPFARA or Change OSPF Area (CHGOSPFARA) command:

```
ADDOSPFARA AREA('1.1.1.1') STUB(*YES)
```

Another type of area is called a *totally stubby area*, which receives less routing information than a stub area. It receives only default routes. Totally stubby areas minimize the compute-intensive operations necessary to build routing trees and also minimize the storage requirements for the topology database. To configure an area as a totally stubby area, set the IMPORT parameter to *YES, on the ADDOSPFARA or CHGOSPFARA command:

```
ADDOSPFARA AREA('1.1.1.1') STUB(*YES) IMPORT(*YES)
```

The OSPF *not so stubby area (NSSA)* type is not supported on i5/OS.

Related concepts

Packet types for OSPF

OSPF sends packets to neighbors to establish and maintain adjacencies, send and receive requests, ensure reliable delivery of Link-state advertisements (LSAs) between neighbors, and to describe link-state databases. Link-state databases are generated from all the LSAs that an area router sends and receives. The link-state database is then used to calculate the shortest-path spanning tree, using the Shortest Path First (SPF) algorithm.

OSPF area aggregation

OSPF aggregation combines groups of routes with common addresses into a single routing table entry. i5/OS supports the use of subnet masks to achieve OSPF aggregation.

OSPF aggregation is used to reduce the size of routing tables. To illustrate, consider a configuration with one backbone area and one nonbackbone area:

- Backbone area 0.0.0.0
 - Router 1
 - Router 3
 - Router 4
- Nonbackbone area 1.1.1.1
 - Network segment 9.92.2.0, subnet mask 255.255.255.0
 - Network segment 9.92.1.0, subnet mask 255.255.255.0
 - ABR2, defined between the backbone and this nonbackbone area.

In the configuration, the nonbackbone area 1.1.1.1 has two network segments but by OSPF aggregation, they are advertised as only one segment, 9.92.0.0/16, to the backbone area. The ABR2 is customized to advertise a range to reduce the number of summary-LSAs transported to the backbone area. Using the following Add OSPF Range (ADDOSPFRRNG) command, the new 9.92.0.0/16 range covers both segments, 9.92.2.0 and 9.92.1.0:

```
ADDOSPFRRNG AREA('1.1.1.1') IPADDRNG('9.92.0.0') SUBNETMASK('255.255.0.0')  
ADVERTISE(*YES)
```

Use either the ADDOSPFRRNG or the Change OSPF Range (CHGOSPFRRNG) command, setting the advertise option to *YES.

Setting the ADVERTISE parameter to *YES provides route summarization in LSAs. If a range or route aggregation is not specified, the LSA summaries from the nonbackbone area, that are sent by ABR2, are as follows:

- 9.92.1.0
- 9.92.2.0

By specifying route aggregation and adding a range to ABR2, the LSA summary that is sent about the nonbackbone area is as follows:

- 9.92.0.0

To specify the range 9.92.0.0/16, the subnet mask that is used is different from those shown in the example for each segment. The subnet mask 255.255.0.0 is the one that is used, because it is common to all subnets within 9.92.0.0/16.

Care is required in adding ranges so that new segments are correctly represented by the subnet mask. To illustrate this, consider the addition of a new nonbackbone area, 2.2.2.2, to the example configuration.

- Backbone area 0.0.0.0
 - Router 1
 - Router 3

- Router 4
- Nonbackbone area 1.1.1.1
 - Network segment 9.92.2.0, subnet mask 255.255.255.0
 - Network segment 9.92.1.0, subnet mask 255.255.255.0
 - ABR2
- Nonbackbone area 2.2.2.2
 - Network segment 9.92.5.0, subnet mask 255.255.255.0
 - Network segment 9.92.6.0, subnet mask 255.255.255.0
 - ABR4

An aggregated subnet mask supports a contiguous clustering of subnets in an area behind one or behind a set of ABRs; however, it is important that the representative subnet mask not make disjointed segments. Generally, a disjointed segment means that a segment is local to an area and not visible outside of that area within the aggregation.

In the example, segments 9.92.5.0 and 9.92.6.0 are added, but if an aggregated mask of 255.255.0.0 is continued for the cluster or subnets numbered 9.92.1.0 and 9.92.2.0, the backbone area appears as if 9.92.0.0/16 is behind both ABR2 and ABR4. This is a disjointed segment which means that the subnet is not a continuous section, but is split on opposite sides of the network. Change the aggregated mask to 255.255.252.0 to aggregate the subnets behind ABR2 as 9.92.0.0/22 and the subnets behind ABR4 as 9.92.4.0/22. The real subnet mask in both disjointed areas can remain 255.255.255.0. Use the ADDOSFPRNG command, specifying a different SUBNETMASK:

```
ADDOSFPRNG AREA('1.1.1.1') IPADDRNG('9.92.0.0') SUBNETMASK('255.255.252.0')  
ADVERTISE(*YES)
```

Note: The slash followed by a number, in the IP address, is standard subnet mask notation and is known as *classless interdomain routing (CIDR)*. Using CIDR, route 9.92.0.0/16, for example, means the subnet mask for network 9.92.0.0 has 16 bits set to the value of 1; therefore, the subnet mask in the example is 255.255.0.0. The subnet mask has four octets, each one has 8 bits, but only the first two octets have all their bits as 1. The SUBNETMASK parameter of the ADDOSFPRNG command determines the range of the network that is advertised to the backbone.

Link-state advertisements

As a participant of an OSPF network, the i5/OS router originates one or more link-state advertisements (LSAs) to send routing information about itself, or to receive routing information from neighbors. This information is used to build the link-state database.

OSPF network types are either point-to-point, broadcast, nonbroadcast multiaccess, point-to-multipoint, or virtual links. The following LSAs are supported by i5/OS:

- Router-LSA

The router-LSA describes all intraarea router destinations, interfaces, and their states. It also indicates whether the originating router has a virtual link, is an *autonomous system border router (ASBR)*, or an ABR. This LSA type also indicates whether each link is a point-to-point connection, a connection to a transit network, a connection to a stub area, or a virtual link.

- Network-LSA

The network-LSA describes the designated connection to a broadcast network. It is originated by the designated router and contains the list of all the routers that are adjacent to the designated router.

- Summary-LSA

This LSA is created by an ABR and describes interarea routes. It contains one advertisement for each IP destination that is inside the other areas that are attached to the ABR.

- ASBR summary LSAs

This LSA is created by an ABR and describes all the routes to the AS boundary routers (ASBR) that are outside the area. It contains the IP address of the AS border router.

- External-LSA

An external-LSA is created by the AS boundary router. It describes routes to destinations outside the AS.

The following types of LSAs are supported by i5/OS for IPv6, only:

- Interarea-prefix LSA

This LSA is similar to the summary-LSA. Each interarea-prefix LSA describes a prefix external to the area that is still internal to the autonomous system. The flooding scope for this LSA is an area.

- Interarea router-LSA

This LSA is similar to the ASBR summary LSA. It describes the path to an ASBR destination router that is external to the area but internal to the AS. The flooding scope for this LSA is an area.

- Link-LSA.

The i5/OS originates a separate link-LSA for each attached link that supports two or more routers. The flooding scope for this LSA is link local.

- Intraarea-prefix LSA

This LSA associates a list of IPV6 address prefixes with a transit network by referencing a network-LSA or it associates a list of IPV6 address prefixes with a router by referencing a router-LSA. The flooding scope for this LSA is an area

Use the following commands as examples that check IPv6 LSAs that are generated:

- Display a link-state database for a specific area.

```
DSPOSPF IPVERSION(*IPV6) OPTION(*STATE) STATE(*LSA) LSA('66.66.66.66')
```

- Display all external-LSAs.

```
DSPOSPF IPVERSION(*IPV6) OPTION(*STATE) STATE(*LSA) LSA(*EXTERNAL)
```

Use the following commands as examples to display IPv4 LSA information, that is received by or originated from i5/OS:

- Display a link-state database for a specific area.

```
DSPOSPF IPVERSION(*IPV4) OPTION(*STATE) STATE(*LSA) LSA('1.1.1.1')
```

- Display all external-LSAs.

```
DSPOSPF IPVERSION(*IPV4) OPTION(*STATE) STATE(*LSA) LSA(*EXTERNAL)
```

Related concepts

OSPF for IPv6

IPv4 and IPv6 have different i5/OS OSPF semantics, addressing, and authentication support.

Aging of link-state records

Link-state advertisements (LSAs) are originated by all routers in an OSPF network. The OMPROUTED server has several considerations in aging or managing LSAs in an i5/OS environment.

- A new LSA has an age of 0.
- The age of the record increments by the transmission delay that is configured for the associated interface.
- The age of the record increments for every second it is maintained.
- The maximum age of an LSA is one hour, at which point the OMPROUTED server no longer uses the LSA in route table calculations. The record is eventually removed from the database.
- The OMPROUTED server must retransmit a record at least once every 30 minutes.

Packet types for OSPF

OSPF sends packets to neighbors to establish and maintain adjacencies, send and receive requests, ensure reliable delivery of Link-state advertisements (LSAs) between neighbors, and to describe link-state databases. Link-state databases are generated from all the LSAs that an area router sends and receives. The link-state database is then used to calculate the shortest-path spanning tree, using the Shortest Path First (SPF) algorithm.

The following packet types are supported for i5/OS in an OSPF environment

Hello packet

This packet is sent by the OMPROUTED server to discover OSPF neighbor routers and to establish bidirectional communications with them.

When several systems or routers that run OSPF have interfaces attached to a common network, the Hello protocol determines the *designated router (DR)*. The DR is adjacent to all routers on the network, and its role is to generate and flood the LSAs, on behalf of the network. In a broadcast network, such as Ethernet, having a DR reduces the amount of router protocol traffic that is generated.

The DR is also responsible for maintaining the network topology database that is replicated at all other routers that are within the same OSPF area on the network.

The concept of a DR does not exist for point-to-point connections.

Database description packet

Once the Hello packets are exchanged and two-way communications are established, the i5/OS and other area routers are neighbors. At this point the OMPROUTED server knows with which neighbors it must establish adjacencies and then starts forming OSPF adjacencies. Bringing up an adjacency is the important OSPF protocol function of synchronizing databases between routers. The database description packets are sent using the exchange database protocol. The exchange database protocol exchanges a description of the link-state databases between adjacent partners, using the database description packet.

Link-state update packet

Until the OMPROUTED databases are fully synchronized, they request and exchange more information from adjacent routers using *link-state request* and *link-state updates* packets.

The router or i5/OS whose *router identifier* is numerically higher, assumes the primary role and the other assumes the secondary role. The primary router sends its database descriptions, one at a time. The secondary router acknowledges each one and includes in the acknowledgement its own database descriptions. The records are compared according to the type, advertising router, and link-state ID. A sequence number in the record determines whether the record is newer or older. If the new description indicates that this record is newer than the recipient already has in its database, this description is saved.

Link-state request packet

After all descriptions are received, the neighbors send out database requests for more complete information about the records that were requested. These requests are followed with a flooding of Link-state updates containing the requested information. Each link-state update packet is acknowledged, either explicitly with a *link-state acknowledgment* packet or implicitly in the link-state packets. The routers are fully adjacent when the link-state databases are fully synchronized.

OSPF adjacency states are Down, Init, Attempt, 2-way, Exstart, Exchange, Loading, and Full. On i5/OS, 2-way is represented by *WAY2 and means that the router is fully adjacent only with the DR or the backup designated router (BDR). When the adjacency state is *FULL, this means that many neighbor states might be *WAY2 and therefore an individual router might know about a neighbor but not be fully adjacent with it.

If the database synchronization process is too long, consider setting the database exchange timeout value higher than the default value that is specified by the *inactive router interval*. Set the database exchange

timeout value when adding or changing an interface with the ADDOSPFIFC or CHGOSPFIFC command. If the database exchange timeout is set higher than the default time, the following occurs:

- The database synchronization fails due to insufficient time.
- The neighbor process never stabilizes beyond neighbor State of *WAY2.

Link-state acknowledgment packet

Each newly received LSA must be acknowledged by its recipient to verify its delivery. This is done by sending a link-state acknowledgment packet, which contains one or more acknowledgements. An acknowledgment packet is either sent immediately or delayed, based on a specified time interval.

To display the status of adjacencies, specify IPv4 or IPv6 on the Display OSPF (DSPOSPF) command, and display the state of the OSPF neighbors by setting the STATE parameter:

```
DSPOSPF IPVERSION(*IPV4) OPTION(*STATE) STATE(*NGH)
```

Related concepts

[OSPF routing domain and areas](#)

OSPF routing depends on the relationship that is defined between areas within a routing domain. CL commands are used to define area types and i5/OS routing neighbors.

OSPF for IPv6

IPv4 and IPv6 have different i5/OS OSPF semantics, addressing, and authentication support.

OSPF for IPv6 addressing

The IPv6 addressing semantics were removed from the OSPF packets and from the main LSA types to make the network independent of the protocol. The main changes are these:

- Addressing information is contained only in an LSA. Hello packets do not contain address information. A new field called Interface ID is assigned by the originating router to uniquely identify the interface to the link.
- The router-LSA and network-LSA types no longer contain network addresses. An LSA address is expressed as a prefix and a prefix length instead of as an IP address and network mask.
- In OSPF for IPv6, neighboring routers are always identified by the router ID. In IPv4, they are identified by an IP address.

Point-to-point links are not supported in IPv6.

IPv6 supports router-LSAs, network LSAs, AS external-LSAs, and a set of other, additional LSA types.

Link replaces subnet

The fundamental mechanisms of OSPF, such as flooding, designated router election, area support, and Shortest Path First calculations remain unchanged when using the IPv6 protocol, but some changes occurred in semantics. The IPv6 term *link* replaces the IPv4 term *subnet*.

IPv6 uses the term link to indicate a communication medium over which nodes communicate at the link layer. With IPv6, multiple subnets are in the same single link and systems communicate directly to the other nodes over the same link, even if they do not share the same IPv6 prefix subnet.

Link local addresses

An IPv6 link local address is useful in automatic address configuration because, by definition, it is assigned to a single physical interface. OSPF uses a link local address to discover all of the neighbors that are attached by the same physical interface link. The OMPROUTED server discovers the addresses of all other routers or systems attached to a link local address, and uses these addresses as next hop information. Using the standard *classless interdomain routing* subnet mask notation, the server needs a link local address in the range of FE80/10.

OSPF IPv6 interface packets are sent using the link local unicast address. To configure an OSPF IPv6 interface, use the Add OSPF Interface (ADDOSPFIFC) command, and associate the interface to a line description which has a link local address.

Unlike IPv6, OSPF virtual link packets are sent using a site local, or globally scoped, IP address, instead of using a link local address.

Authentication

OSPF for IPv6 implementation on i5/OS does not support authentication because authentication is already included in IPv6 packet support. The fields for the authentication in OSPF for IPv6 are removed from the OSPF packet headers. When adding or changing an OSPF IPv6 area or interface, all authentication fields are ignored.

Related concepts

[i5/OS OSPF Authentication](#)

Configure the TCP/IP server to authenticate i5/OS in an OSPF environment.

[Link-state advertisements](#)

As a participant of an OSPF network, the i5/OS router originates one or more link-state advertisements (LSAs) to send routing information about itself, or to receive routing information from neighbors. This information is used to build the link-state database.

Related information

[IPv6 address types](#)

OSPF interfaces

For i5/OS, an OSPF interface is an Internet address that represents a logical TCP/IP connection that is associated with an existing i5/OS line description. The line description is configured as an OSPF interface.

Use the ADDOSPFIFC command to set the different parameters needed by the OMPROUTED server to identify an interface as a usable OSPF routing interface. For i5/OS, the interface identifier is an IPv4 or IPv6 address.

OSPF uses multicast functions when interfaces are attached to broadcast networks, such as Ethernet. For all OSPF routers, these interfaces are joined to multicast address 224.0.0.5. Only the packets that are destined for the 224.0.0.5 IP address are received by interfaces that are joined to the multicast IP address. IPv6 uses ff02::5 for the multicast address.

Note: Do not define neighbors on nonbroadcast or multicast-capable media because the OMPROUTED server communicates OSPF information only to those neighbors that are defined and cannot form adjacencies with any additional neighbors.

To identify i5/OS TCP/IP interfaces that are valid OSPF interfaces, use NETSTAT OPTION(*IFC) for IPv4, or use NETSTAT OPTION(*IFC6) for IPv6. If multiple OSPF interface identifiers that belong to a common subnet are added to an OSPF configuration, only one interface actually exchanges OSPF information.

Related information

[Add OSPF Interface \(ADDOSPFIFC\)](#)

[Remove OSPF Interface \(RMVOSPFIFC\)](#)

[Change OSPF Interface \(CHGOSPFIFC\)](#)

Point-to-point links for OSPF

A point-to-point connection is a serial link that connects two routers, making them Open Shortest Path First (OSPF) neighbors. Point-to-point connections are supported for i5/OS in an OSPF environment.

For i5/OS to discover neighbors over point-to-point links, a connection profile or a line description that represents a connection to a modem resource is required. The modem is in leased or switched mode.

Use the Add OSPF Interface (ADDOSPFIFC) command to add point-to-point interfaces to an OSPF configuration by specifying the IFC and PPPCNNPRF parameters, where *connection_profile_name* represents an existing profile in the connection profiles list:

```
ADDOSPFIFC IFC(*PPPCNNPRF) PPPCNNPRF(connection_profile_name)
```

It is not required that a connection profile is active when a point-to-point connection is added to a configuration. If the local interface that is eventually used to send OSPF packets over a point-to-point link is not known, the interface status can show *DOWN. When the profile becomes active, it informs the OMPROUTED server to use the interface that sends the packets.

Related information

[Remote Access Services: PPP connections](#)

i5/OS OSPF Authentication

Configure the TCP/IP server to authenticate i5/OS in an OSPF environment.

OSPF authentication of a system controls the passing of packets in an OSPF networking environment. OSPF authentication is controlled by the type that is specified when adding or changing an OSPF interface or an OSPF virtual link, using the AUTHTYPE parameter on either of the following commands:

- Add OSPF Interface (ADDOSPFIFC)
- Change OSPF Interface (CHGOSPFIFC)
- Add OSPF Virtual Link (ADDOSPFLNK)
- Change OSPF Virtual Link (CHGOSPFLNK)

If AUTHTYPE is not specified on one of the preceding commands, it is inherited from the type defined for the OSPF area, that is specified in the command.

The AUTHTYPE parameter is ignored for OSPF IPv6 interfaces.

Configuring the OMPROUTED TCP/IP server to authenticate to a system that is participating in an OSPF network has the following advantages:

- Changes are limited to only trusted systems or routers.
- Protection is provided against the accidental introduction of a router that has the potential of introducing malicious, alien routers into the OSPF network.

The OMPROUTED server supports both Message-Digest algorithm 5 (MD5) authentication and simple password authentication.

Message-Digest algorithm 5 (MD5) authentication

On the i5/OS platform, an MD5 value is expressed as a 32-character hexadecimal number or as an ASCII value.

To supply a hexadecimal value for MD5 authentication, follow these requirements:

- The length must be exactly 32 characters, and the first character must be an **X**.
- The entire authentication key must be enclosed in single quotation marks.

This is an example of supplying a hexadecimal value for MD5 authentication:

```
ADDOSPFIFC IFC('9.7.85.1') AREA('1.1.1.1') AUTHTYPE(*MD5) AUTHVAL (124  
'Xffffffffffffffffffffffffffff')
```

To supply an ASCII value for MD5 authentication, follow these requirements:

- The key must be Cisco compatible.
- The maximum length is 16 characters, and the first character must be an **A** or an **a**.
- The entire authentication key must be enclosed in single quotation marks.

This is an example of supplying an ASCII value for MD5 authentication:

```
ADDOSPFIFC IFC('9.7.85.1') AREA('1.1.1.1') AUTHTYPE(*MD5) AUTHVAL(124 'aACDZ')
```

Simple password authentication

On the i5/OS platform, a simple password value is expressed as a 16-character hexadecimal number or as an ASCII value.

To supply a hexadecimal value for simple password authentication, follow these requirements:

- The length must be exactly 16 characters, and the first character must be an **X**.
- The entire authentication key must be enclosed in single quotation marks.

This is an example of supplying a hexadecimal value simple password authentication:

```
ADDOSPFIFC IFC('9.7.85.1') AREA('1.1.1.1') AUTHTYPE(*PASSWORD) AUTHVAL(124 'Xfffffffffffffff')
```

To supply an ASCII value for simple password authentication, follow these requirements:

- The key must be Cisco compatible.
- The maximum length is 8 characters.
- The entire authentication key must be enclosed in single quotation marks.

This is an example of supplying an ASCII value for simple password authentication:

```
ADDOSPFLNK NGHRTR('2.2.2.2') LNKTMSARA('1.1.1.1') AUTHTYPE(*PASSWORD) AUTHVAL(*N  
'del8gado')
```

Related concepts

OSPF for IPv6

IPv4 and IPv6 have different i5/OS OSPF semantics, addressing, and authentication support.

Enabling of i5/OS OSPF job tracing

Messages are logged in the associated i5/OS job log, to trace the status of OSPF routing.

During OSPF routing, messages are sent to the QTOOROUTE job log to inform the user of the following:

- The state changes of the configured OSPF interfaces, when the Hello protocol is processing.
- The state changes of the discovered neighbors, during the database exchange process.
- The state of routes discovered by OSPF.

Trace the QTOOROUTE job to verify the flow of the OSPF protocol. The trace displays the packets that are sent or received over the OSPF interfaces.

To enable the trace for OSPF, set the TRCTYPE to *ROUTING and, optionally, set the other trace levels. The following example uses the Start Trace (STRTRC) command to start a trace for OSPF, filtering only the traces for OSPF:

```
STRTRC SSNID(QTOOROUTE) JOB(( *ALL/QTCP/QTOOROUTE)) JOBTRCTYPE(*TRCTYPE)  
TRCTYPE(( *ROUTING *VERBOSE))
```

Tracing is supported for the OSPF CL commands by tracing an active i5/OS session. Use the following as an example:

```
STRTRC SSNID(OSPF) JOBTRCTYPE(*TRCTYPE) TRCTYPE(( *ROUTING *VERBOSE))
```

Open Shortest Path First support tasks

This topic provides the steps needed to configure your i5/OS to participate in an OSPF network. It also identifies the different commands that you can use to start, end, and restart a TCT/IP interface.

Configuring i5/OS for OSPF networking

An OSPF configuration includes a router identifier, an area, and OSPF registered interfaces that attach within the area. i5/OS provides a set of CL commands to establish it as a participant in an OSPF network.

To identify i5/OS TCP/IP interfaces that are valid OSPF interfaces, use NETSTAT OPTION(*IFC) for IPv4, or use NETSTAT OPTION(*IFC6) for IPv6. See the Enabling TCP/IP for OSPF on i5/OS topic for starting and stopping TCP/IP information.

Follow these steps to configure i5/OS to participate in an OSPF network.

1. Change OSPF Attributes (CHGOSPFA)

to establish a router identifier.

The CHGOSPFA command establishes the router identifier which is the unique, system identifier in an OSPF network. It is an IP address in the dotted-decimal format. A CHGOSPFA example is:

```
CHGOSPFA ROUTER('1.1.1.1')
```

2. Add OSPF Area (ADDOSPFARA)

to configure the area identifier.

The ADDOSPFARA command identifies the OSPF area in an intranet or autonomous system (AS) where the OSPF runs. It is a collection of IP networks in which the OSPF interfaces attach. The area must exist when interfaces are added to the OSPF configuration. The area identifier is represented in dotted-decimal format. ADDOSPFARA examples are:

```
For IPv4, ADDOSPFARA AREA('1.1.1.1')
```

```
For IPv6, ADDOSPFARA ('1.1.1.1') IPVERSION(*IPV6)
```

3. Add OSPF Interface (ADDOSPFIFC)

to configure the interface identifier.

An OSPF interface is an Internet address that represents a configured logical TCP/IP interface, and is associated with a line description that exists on a system. Use the ADDOSPFIFC command to set the different parameters needed by the OMPROUTED server to add the interface into the routing protocol implementation. ADDOSPFIFC examples are:

```
For IPv4, ADDOSPFIFC IFC('9.7.85.1') AREA('1.1.1.1')
```

```
For IPv6, ADDOSPFIFC IFC('2000::38') AREA('1.1.1.1')
```

Related concepts

[Scenario: Configuring OSPF interfaces and neighbors](#)

This scenario shows an OSPF configuration of i5/OS systems in a sample network.

Related information

[Change OSPF Attributes \(CHGOSPFA\)](#)

[Add OSPF Area \(ADDOSPFARA\)](#)

[Add OSPF Interface \(ADDOSPFIFC\)](#)

Enabling TCP/IP for OSPF on i5/OS

It is a requirement that the i5/OS TCP/IP interface is active before using OSPF.

1. Use the various options of the Configure TCP/IP (CFGTCP) command to establish a TCP/IP OSPF interface.
2. Use the Start TCP/IP Server (STRTCPSVR) command to start the server:

```
STRTCPSVR SERVER(*OMPROUTED) INSTANCE(*OSPF)
```


This command submits the QTOOROUTE job in the QSYSWRK subsystem. If the command is issued while the QTOOROUTE job is active, a message is sent to the user job log. Specifying the INSTANCE option starts only the OSPF routing function. You can optionally omit this parameter.

Once the TCP/IP interface is active, use the following information to end the server or to restart the server, as needed.

- End an OSPF routing function, using the following command:

ENDTCPSVR SERVER(*OMPROUTED) INSTANCE(*OSPF)

This command ends only the OSPF function from the QTOOROUTE job. The job is still active under the QSYSWRK subsystem.

- Restart the OMPROUTED server using the following command:

STRTCPSVR SERVER(*OMPROUTED) RESTART(*OMPROUTED)

This command resubmits the QTOOROUTE job in the QSYSWRK subsystem. It reprocesses both the configuration and the index file to synchronize them. Unpredictable results occurs if these files are not synchronized. Importing only the config file from another system, for example, can cause the files to be desynchronized, and require that the server is restarted.

Related information

[TCP/IP setup](#)

[Open Shortest Path First](#)

[End TCP/IP Server \(ENDTCPSVR\)](#)

[Start TCP/IP Server \(STRTCPSVR\)](#)

Open Shortest Path First support reference

For your reference, the i5/OS API and CL commands that are used in OSPF routing are covered in this topic. The topic also includes scenarios that you can reference in establishing your OSPF routing environment.

Open Shortest Path First API and commands

Use control language (CL) commands to manage OSPF attributes, interfaces, areas, ranges, links, statistics, state information, and to log OSPF activity. Use the Retrieve OSPF State Information API to retrieve OSPF statistics.

CL commands

The Configure Routing Protocols (CFGRTG) command displays a list of all the OSPF commands on a system.

<i>Table 1. CL commands and supported functions</i>		
CL name	CL command	Function
Change OSPF Attributes	CHGOSPFA	Set general OSPF attributes.
Display OSPF	DSPOSPF	Show the current OSPF configuration or the state of the routing protocol.
Add OSPF Interface	ADDOSPFIFC	Add interfaces for OSPF.
Remove OSPF Interface	RMVOSPFIFC	Remove interfaces for OSPF.
Change OSPF Interface	CHGOSPFIFC	Change interfaces for OSPF.
Add OSPF Area	ADDOSPFARA	Add areas for OSPF.
Remove OSPF Area	RMVOSPFARA	Remove areas for OSPF.

Table 1. CL commands and supported functions (continued)

CL name	CL command	Function
Change OSPF Area	CHGOSPFARA	Change areas for OSPF.
Add OSPF Range	ADDOSPFRNG	Add ranges for OSPF.
Remove OSPF Range	RMVOSPFRNG	Remove ranges for OSPF.
Change OSPF Range	CHGOSPFRNG	Change ranges for OSPF.
Add OSPF Virtual Link	ADDOSPFLNK	Add virtual links for OSPF.
Remove OSPF Virtual Link	RMVOSPFLNK	Remove virtual links for OSPF.
Change OSPF Virtual Link	CHGOSPFLNK	Change virtual links for OSPF.

Display OSPF command

The Display OSPF (DSPOSPF) command displays the OSPF configuration that is set in the configuration file. The OSPF version, IPv4 or IPv6, is a required parameter for the DSPOSPF command.

The OPTION parameter specifies whether configuration or state information is displayed. To display state information, which is the status and runtime statistics of the interfaces, set the OPTION parameter to *STATE:

```
DSPOSPF IPVERSION(*IPV4) OPTION(*STATE)
```

To display configuration information, set the OPTION parameter to *CFG and then specify the type of configuration information that is displayed by setting the CONFIG parameter. To retrieve area, interface, neighbor, or virtual link information, specify one of these for the CONFIG parameter. To display the router identifier and the OSPF routing protocol status, specify *GLOBAL for the CONFIG parameter.

DSPOSPF IPVERSION(*IPV4) OPTION(*CFG) CONFIG(*GLOBAL)

Note:

To specify *GLOBAL, the Simple Network Management Protocol (SNMP) server must be started. Type the following commands to start or verify that the server is active:

- To start the server, use the Start TCP/IP Server (STRTCPSVR) command, specifying the SNMP server:

```
STRTCPSVR *SNMP
```

- To verify the server is active, use the WRKJOB command, specifying the QTMSNMP job name, and then selecting option 1:

```
WRKJOB QTMSNMP
```

Retrieve Open Shortest Path First State Information API

The Retrieve Open Shortest Path First State Information (QtooRtvOSPFdta) API retrieves the Open Shortest Path First (OSPF) statistics from the OMPROUTED TCP server.

Related concepts

Scenario: Retrieve OSPF State Information API

This scenario uses the i5/OS Retrieve OSPF State Information (QtooRtvOSPFdta) API to retrieve OMPROUTED server information.

Related information

[Retrieve OSPF State Information \(QtooRtvOSPFdta\) API](#)

[Change OSPF Attributes \(CHGOSPFA\)](#)

[Add OSPF Area \(ADDOSPFARA\)](#)

[Add OSPF Interface \(ADDOSPFIFC\)](#)

[Add OSPF Virtual Link \(ADDOSPFLNK\)](#)

[Add OSPF Range \(ADDOSPFRNG\)](#)
[Change OSPF Area \(CHGOSPFARA\)](#)
[Change OSPF Interface \(CHGOSPFIFC\)](#)
[Change OSPF Virtual Link \(CHGOSPFLNK\)](#)
[Change OSPF Range \(CHGOSPFRNG\)](#)
[Display OSPF \(DSPOSPF\)](#)
[End TCP/IP Server \(ENDTCPSVR\)](#)
[Remove OSPF Area \(RMVOSPFARA\)](#)
[Remove OSPF Interface \(RMVOSPFIFC\)](#)
[Remove OSPF Virtual Link \(RMVOSPFLNK\)](#)
[Remove OSPF Range \(RMVOSPFRNG\)](#)
[Start TCP/IP Server \(STRTCPSVR\)](#)
[Start Trace \(STRTRC\)](#)

Scenarios: Configuring OSPF

One scenario shows OSPF configuration entries in a sample network, one demonstrates OSPF multipath routes, and another demonstrates the i5/OS OSPF API.

Note: By using the code examples, you agree to the terms of the [“Code license and disclaimer information”](#) on page 24.

Scenario: Configuring OSPF interfaces and neighbors

This scenario shows an OSPF configuration of i5/OS systems in a sample network.

The sample configuration shows the interfaces and neighbor state of System A and System B. The Hello protocol and database exchange protocols have discovered the neighbor routers and formed adjacencies between them. The OSPF protocol maintains the link-state database synchronization in both routers.

Note: By using the code examples, you agree to the terms of the [“Code license and disclaimer information”](#) on page 24.

Figure 2. OMPROUTED state

System A was configured using the following steps and information:

1. The router identifier was set, using the Change OSPF Attributes (CHGOSPFA) command:
`CHGOSPFA ROUTER('1.1.1.1')`
2. The OSPF area identifier was added, using the Add OSPF Area (ADDOSPFARA) command:
`ADDOSPFARA AREA('1.1.1.1')`
3. The OSPF interface was configured, using the Add OSPF Interface (ADDOSPFIFC) command, after verifying that the interface 9.7.85.1 was a valid TCP/IP interface:
`ADDOSPFIFC IFC('9.7.85.1') AREA('1.1.1.1')`

System B was configured using the following steps and information:

1. The router identifier was set, using the CHGOSPFA command:
`CHGOSPFA ROUTER('2.2.2.2')`
2. The OSPF area identifier was added, using the ADDOSPFARA command:
`ADDOSPFARA AREA('2.2.2.2')`
3. The OSPF interface was configured, using the ADDOSPFIFC command, after verifying that the interface 9.7.85.2 was a valid TCP/IP interface:
`ADDOSPFIFC IFC('9.7.85.2') AREA('2.2.2.2')`

Server status

The Display OSPF Global Information command was used to display the OSPF protocol status, which is the status of the OMPROUTED server that runs the OSPF protocol. The Start TCP/IP Server (STRTCPSVR) command was used to change the status from disabled to enabled for the remainder of this example.

```
Display OSPF Global Information
System: MEXGPL21
OSPF protocol status . . . . . : DISABLED
Router identifier . . . . . : 1.1.1.21
Boundary capability . . . . . : ENABLED
Area border router capability . . : DISABLED
Autostart server . . . . . : *YES
Subagent . . . . . : *YES
```

Figure 3. Displayed OSPF Global Information to see the server status.

Interface status

After the OMPROUTED server job was started on both systems, the Display OSPF (DSPOSPF) command was used to verify the state of each OSPF interface. Status was retrieved for both System A and System B, by specifying IPv4 as the IP version on the command:

DSPOSPF IPVERSION(*IPV4) OPTION(*STATE) STATE(*IFC)

```
Display OSPF State of Configured Interfaces
System: RCHLP610
Router identifier . . . . . : 1.1.1.1
Number of configured interfaces . . . : 1

Configured interface list
Type options, press Enter.
5=Display details

Opt  Interface      Area      Interface  Interface  Number
     Identifier    Identifier Type      Status    of
     9.5.85        1.1.1.1  *BCST     *BDR      Neighbors
                                     1

F3=Exit  F5=Refresh  F6=Print  F12=Cancel

Bottom
```

Figure 4. Verified the state of OSPF interface of System A.

```

Display OSPF State of Configured Interfaces
Router identifier . . . . . : 2.2.2.2
Number of configured interfaces . . . : 1
System: RCHLP611

Configured interface list

Type options, press Enter.
5=Display details

Opt  Interface      Area      Interface  Interface  Number
     Identifier    Identifier Type      Status     of
     9.7.85.2      1.1.1.1  *BCST     *DR        Neighbors
                                           1

F3=Exit  F5=Refresh  F6=Print  F12=Cancel

Bottom

```

Figure 5. Verified the state of OSPF interface of System B.

Link-state advertisements

The link-state database contained three link-state advertisements (LSAs). The following command was used to display the contents of the database for each system. All routers and System i® nodes that belong to the same OSPF area have the same content in the link-state database.

DSPOSPF IPV4(*IPV4) OPTION(*STATE) STATE(*LSA) LSA('1.1.1.1')

```

Display Link State Advertisements
Number of link state advertisements . : 3
Total checksum . . . . . : 1e561
System: RCHLP610

Link state advertisement list

Link State Type  Link State Destination  Link State Originator  Link State Sequence Number  Link State Age  Link State Checksum
*LSTRL 1.1.1.1 1.1.1.1 80000024 1270 91ef
*LSTRL 2.2.2.2 2.2.2.2 80000086 1275 cd4d
*LSTNL 2.2.2.2 2.2.2.2 8000001e 820 8625

```

Figure 6. Displayed LSAs for System A.

```

Display Link State Advertisements
Number of link state advertisements . . : 3
Total checksum . . . . . : 1e561
System: RCHLP610

Link state advertisement list

Link State Type      Link State Destination  Link State Originator  Link State Sequence Number  Link State Age  Link State Checksum
*LSTRL 1.1.1.1          1.1.1.1          80000024    1270        91ef
*LSTRL 2.2.2.2          2.2.2.2          80000086    1275        cd4d
*LSTNL 2.2.2.2          2.2.2.2          8000001e    820         8625

```

Figure 7. Displayed LSAs for System B.

Related tasks

Configuring i5/OS for OSPF networking

An OSPF configuration includes a router identifier, an area, and OSPF registered interfaces that attach within the area. i5/OS provides a set of CL commands to establish it as a participant in an OSPF network.

Scenario: OSPF multipath routes

This scenario demonstrates using different OSPF routes to the same i5/OS. An advantage of OSPF is that it can calculate multipath, equal-cost routes to the same destination or system.

Sample configuration

This example has four physical System i units. Systems R1, R2, and R4 have OSPF interfaces attached to the same Ethernet network (10.1.1.1). Systems R2 and R4 have OSPF interfaces type point-to-point connections to system R3. Each interface has a cost of one and all systems are in the same OSPF area (10.10.10.10).

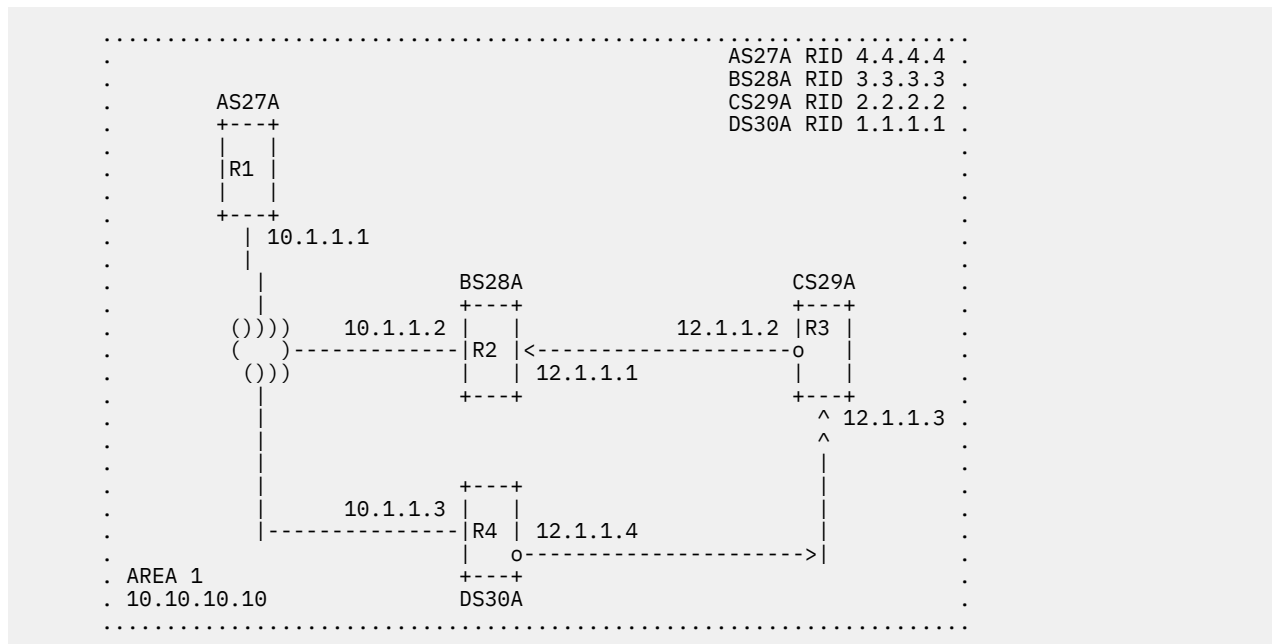


Figure 8. A multipath configuration

Retrieved TCP/IP route information

To display routing information, Option 2, which is Display TCP/IP route information, was selected after typing the following NETSTAT command on the i5/OS command line:

```
NETSTAT *RTE
```

The following information was revealed in the displays that follow:

- Two duplicate routes to 12.1.1.4, one by next hop 10.1.1.2 and the other one by 10.1.1.3.
- Two duplicate routes to 12.1.1.1, which have paths through two different next hops, one by 10.1.1.2 and other one by 10.1.1.3.

```

Display TCP/IP Route Information
System: AS27A
Type options, press Enter.
5=Display details

Opt Route Subnet Next Route
Destination Mask Hop Available
9.5.130.0 255.255.255.0 *DIRECT *YES
10.1.1.0 255.255.255.0 *DIRECT *YES
12.1.1.4 *HOST 10.1.1.2 *YES
12.1.1.4 *HOST 10.1.1.3 *YES
12.1.1.3 *HOST 10.1.1.3 *YES
12.1.1.2 *HOST 10.1.1.2 *YES
12.1.1.1 *HOST 10.1.1.2 *YES
12.1.1.1 *HOST 10.1.1.3 *YES
127.0.0.0 255.0.0.0 *DIRECT *YES
224.0.0.0 240.0.0.0 *DIRECT *YES
224.0.0.0 240.0.0.0 *DIRECT *YES
224.0.0.0 240.0.0.0 *DIRECT *YES
*DFTRROUTE *NONE 9.5.130.1 *YES

F3=Exit F5=Refresh F9=Command line F11=Display route type F12=Cancel
F13=Sort by column F20=Display IPv6 routes F24=More keys
Bottom

```

Figure 9. The NETSTAT command retrieved routing information

```

Display TCP/IP Route Information
System: AS27A
Type options, press Enter.
5=Display details

Opt Route Type of Route Route Route
Destination Service MTU Type Source
9.5.130.0 *NORMAL 4100 *DIRECT *CFG
10.1.1.0 *NORMAL 1492 *DIRECT *CFG
12.1.1.4 *NORMAL 1492 *HOST *OSPF
12.1.1.4 *NORMAL 1492 *HOST *OSPF
12.1.1.3 *NORMAL 1492 *HOST *OSPF
12.1.1.2 *NORMAL 1492 *HOST *OSPF
12.1.1.1 *NORMAL 1492 *HOST *OSPF
12.1.1.1 *NORMAL 1492 *HOST *OSPF
127.0.0.0 *NORMAL 576 *DIRECT *CFG
224.0.0.0 *NORMAL 1492 *DIRECT *CFG
224.0.0.0 *NORMAL 4100 *DIRECT *CFG
224.0.0.0 *NORMAL 576 *DIRECT *CFG
*DFTRROUTE *NORMAL 4100 *DFTRROUTE *CFG

F3=Exit F5=Refresh F9=Command line F11=Display route type F12=Cancel
F13=Sort by column F20=Display IPv6 routes F24=More keys
Bottom

```

Figure 10. The NETSTAT command retrieved routing information

Scenario: Retrieve OSPF State Information API

This scenario uses the i5/OS Retrieve OSPF State Information (QtooRtvOSPFdta) API to retrieve OMPROUTED server information.

Note: By using the code examples, you agree to the terms of the [“Code license and disclaimer information”](#) on page 24.

```
/** START HEADER FILE SPECIFICATIONS *****/
/*
/* Source File Name: H/OSPFAPITEST
/*
/* Descriptive Name: Retrieve OSPF State Information
/*
/* Description: The Retrieve OSPF State Information(QtooRtvOSPFdta) */
/*               API retrieves information about OSPF that runs
/*               under the OMPROUTED server.
/*
/* Header Files Included: None.
/*
/* Macros List: None.
/*
/* Structure List:
/*Qtoo_SPFI0100_t           (OSPF General info)
/*Qtoo_IPv4_OSPF_Area_Entry_t (OSPF IPv4 Area List Entry)
/*Qtoo_IPv6_OSPF_Area_Entry_t (OSPF IPv6 Area List Entry)
/*Qtoo_IPv4_OSPF_Ifc_Entry_t (OSPF IPv4 Interface Entry)
/*Qtoo_IPv6_OSPF_Ifc_Entry_t (OSPF IPv6 Interface Entry)
/*Qtoo_IPv4_OSPF_Neighbor_Entry_t (OSPF IPv4 Neighbor Entry)
/*Qtoo_IPv6_OSPF_Neighbor_Entry_t (OSPF IPv6 Neighbor Entry)
/*Qtoo_IPv4_OSPF_Vtl_Link_Entry_t (OSPF IPv4 Virtual Link Entry)
/*Qtoo_IPv6_OSPF_Vtl_Link_Entry_t (OSPF IPv6 Virtual Link Entry)
/*Qtoo_IPv4_OSPF_LSA_Entry_t (OSPF IPv4 LSA Entry)
/*Qtoo_IPv6_OSPF_LSA_Entry_t (OSPF IPv6 LSA Entry)

#include <QSYSINC/H/QT00SPF1>
#include <stdio.h>
#include <signal.h>
#include <string.h>
#include <stdlib.h>
#include <quscrtuq.h>
#include <qusdltug.h>
#include <qusec.h>

/***** Structures *****/
typedef struct {
    Qus_EC_t ec_fields;
    char    exception_data[100];
} error_code_struct;

/***** Print Main (General) Info *****/
void prtMainInfo(Qtoo_SPFI0100_t *OSPF_Ptr)
{
    printf("Router Id: %s \n", OSPF_Ptr->RouterID);
    printf("Attached Areas: %d \n", OSPF_Ptr->Attached_Areas);
    printf("Dijkstra Runs: %d \n", OSPF_Ptr->Dijkstra_Runs);
    printf("Packages Received: %d \n", OSPF_Ptr->OSPFpkt_Received);
    printf("LSA's Allocated: %d \n", OSPF_Ptr->LSA_Allocated);
    printf("Number IPv4 Areas: %d \n", OSPF_Ptr->Number_Of_IPv4_Area_Lst_Ent);
    printf("Number IPv6 Areas: %d \n", OSPF_Ptr->Number_Of_IPv6_Area_Lst_Ent);
    printf("Number IPv4 Interfaces: %d \n", OSPF_Ptr->Number_Of_IPv4_Ifc_Lst_Ent);
    printf("Number IPv6 Interfaces: %d \n", OSPF_Ptr->Number_Of_IPv6_Ifc_Lst_Ent);
    printf("Number IPv4 Neighbors: %d \n", OSPF_Ptr->Number_Of_IPv4_Neigh_Lst_Ent);
    printf("Number IPv6 Neighbors: %d \n", OSPF_Ptr->Number_Of_IPv6_Neigh_Lst_Ent);
    printf("Number IPv4 Virtual Links: %d \n", OSPF_Ptr->Number_Of_IPv4_Vtl_Lst_Ent);
    printf("Number IPv6 Virtual Links: %d \n", OSPF_Ptr->Number_Of_IPv6_Vtl_Lst_Ent);
    printf("Number IPv4 LSA's: %d \n", OSPF_Ptr->Number_Of_IPv4_Vtl_Lst_Ent);
    printf("Number IPv6 LSA's: %d \n", OSPF_Ptr->Number_Of_IPv6_Vtl_Lst_Ent);
}

/***** Main *****/
int main ()
{
    /*Reserving the enough bytes for the header in the ospf structure*/
    char* OSPF_Receiver = (char*)malloc (1000);
    int Length_Receiver = 1000;
    int i;
    error_code_struct error_code;
}
```



```

/*****
/* Initialize the error code parameter. */
/*****/

error_code.ec_fields.Bytes_Provided=sizeof(error_code_struct);

/*****
/* Prototype for calling Retrieve OSPF State Information API */
/* (QtooRtvOSPFData) */
/*****/

QtooRtvOSPFData( OSPF_Receiver, /* Receiver variable */
                &Length_Receiver, /* Length of receiver variable */
                "SPFI0100", /* Format name */
                (char*)&error_code); /* Error code */

/*****
/* If an exception occurred, the API would have returned the */
/* exception in the error code parameter. The bytes available */
/* field will be set to zero if no exception occurred and greater */
/* than zero if an exception did occur. */
/*****/

if (error_code.ec_fields.Bytes_Available > 0)
{
    printf("FAILED WITH EXCEPTION:%s",
          error_code.ec_fields.Exception_Id);
    exit(1);
}

Qtoo_SPFI0100_t *OSPF_Ptr = (Qtoo_SPFI0100_t *) OSPF_Receiver;

if( OSPF_Ptr->Bytes_Returned < OSPF_Ptr->Bytes_Available)
{
    printf("Recalculating space for receiver");
    /*Recalculating the enough bytes for the header in the ospf structure*/
    OSPF_Receiver = (char*)realloc (OSPF_Receiver,OSPF_Ptr->Bytes_Available);
}

/*****
/* Prototype for calling Retrieve OSPF State Information API */
/* (QtooRtvOSPFData) */
/*****/

QtooRtvOSPFData( OSPF_Receiver, /* Receiver variable */
                Length_Receiver, /* Length of receiver variable */
                "SPFI0100", /* Format name */
                (char *)&error_code); /* Error code */

/*Qtoo_SPFI0100_t *OSPF_Ptr = (Qtoo_SPFI0100_t *) OSPF_Receiver;*/
OSPF_Ptr = (Qtoo_SPFI0100_t *) OSPF_Receiver;
}

/* print general info */
printf("Router Id: %s \n", OSPF_Ptr->RouterID);
prtMainInfo(OSPF_Ptr);
printf("Router Id: %s \n", OSPF_Ptr->RouterID);

char* IPv4_Ptr = (char*) OSPF_Receiver;

/*Cast of the Qtoo_IPv4_OSPF_Area_Entry_t structure is made to the IPv4_Ospf_Area_Ptr */
/*pointer and the displacement will be Offset_To_IPv4_Area_Lst bytes, to aim at the */
/*Offset_To_IPv4_Area_Lst of the list in the structure */

Qtoo_IPv4_OSPF_Area_Entry_t *IPv4_Ospf_Area_Ptr=(Qtoo_IPv4_OSPF_Area_Entry_t*)(IPv4_Ptr+OSPF_Ptr-
>Offset_To_IPv4_Area_Lst);
for(unsigned long i=0;i < OSPF_Ptr->Number_Of_IPv4_Area_Lst_Ent;i++)
{
    printf("\nPrinting IPv4 Area Number %d",i);
    printf("\nArea ID: %s",IPv4_Ospf_Area_Ptr->AreaID);
    printf("\nOSPF_Runs: %d",IPv4_Ospf_Area_Ptr->SPF_Runs);
    printf("\nArea_BR_Count: %d",IPv4_Ospf_Area_Ptr->Area_BR_Count);
    printf("\nTotal_Number_of_LSA_Area: %d",IPv4_Ospf_Area_Ptr->Total_Number_of_LSA_Area);
    /*Increment IPv4_Ospf_Area_Ptr bytes in order to point to the next structure*/
    IPv4_Ospf_Area_Ptr+=sizeof(Qtoo_IPv4_OSPF_Area_Entry);
}

char *IPv6_Ptr = (char*)(OSPF_Receiver);

/*Cast of the Qtoo_IPv6_OSPF_Area_Entry_t structure is made to the IPv6_Ospf_Area_Ptr */
/*pointer and the displacement will be Offset_To_IPv4_Area_Lst bytes, to aim at the */
/*Offset_To_IPv6_Area_Lst of the list in the structure */

Qtoo_IPv6_OSPF_Area_Entry_t *IPv6_Ospf_Area_Ptr=(Qtoo_IPv6_OSPF_Area_Entry_t*)(IPv6_Ptr+OSPF_Ptr-
>Offset_To_IPv6_Area_Lst);
for(unsigned long i=0;i < OSPF_Ptr->Number_Of_IPv6_Area_Lst_Ent;i++)
{
    printf("\nPrinting IPv6 Area Number %d",i);
    printf("\nArea ID: %s",IPv6_Ospf_Area_Ptr->AreaID);
    printf("\nOSPF_Runs: %d",IPv6_Ospf_Area_Ptr->SPF_Runs);
    printf("\nArea_BR_Count: %d",IPv6_Ospf_Area_Ptr->Area_BR_Count);
    printf("\nTotal_Number_of_LSA_Area: %d",IPv6_Ospf_Area_Ptr->Total_Number_of_LSA_Area);
    /*Increment IPv6_Ospf_Area_Ptr bytes in order to point to the next structure*/
}

```

```

    IPv6_Ospf_Area_Ptr+=sizeof(Qtoo_IPv6_OSPF_Area_Entry);
}

char *IPv4_Ifc_Ptr = (char*)(OSPF_Receiver);

/*Cast of the Qtoo_IPv4_OSPF_Ifc_Entry_t structure is made to the IPv4_Ifc_Entry_Ptr */
/*pointer and the displacement will be Offset_To_IPv4_Ifc_Lst bytes, to aim at the */
/*Offset_To_IPv4_Ifc_Lst of the list in the structure */

Qtoo_IPv4_OSPF_Ifc_Entry_t *IPv4_Ifc_Entry_Ptr = (Qtoo_IPv4_OSPF_Ifc_Entry_t*)(IPv4_Ifc_Ptr+OSPF_Ptr-
>Offset_To_IPv4_Ifc_Lst);
for(unsigned long i=0;i < OSPF_Ptr->Number_Of_IPv4_Ifc_Lst_Ent;i++)
{
    printf("\nPrinting IPv4 Interface Entry %d",i);
    printf("\nIp_Address: %s",IPv4_Ifc_Entry_Ptr->IP_Address);
    printf("\nArea_ID: %s",IPv4_Ospf_Area_Ptr->AreaID);
    printf("\nInterface_Type: %d",IPv4_Ifc_Entry_Ptr->Interface_Type);
    printf("\nDesignated_Router_Pri: %d",IPv4_Ifc_Entry_Ptr->Designated_Router_Pri);
    printf("\nTransmission_Dly: %d",IPv4_Ifc_Entry_Ptr->Transmission_Dly);
    printf("\nRetransmission_Dly: %d",IPv4_Ifc_Entry_Ptr->Retransmission_Dly);
    printf("\nHello_Interval: %d",IPv4_Ifc_Entry_Ptr->Hello_Interval);
    printf("\nInactive_Router_Interval: %d",IPv4_Ifc_Entry_Ptr->Inactive_Router_Interval);
    printf("\nPoll_Interval: %d",IPv4_Ifc_Entry_Ptr->Poll_Interval);
    printf("\nInterface_State: %d",IPv4_Ifc_Entry_Ptr->Interface_State);
    printf("\nDesignated_Router: %d",IPv4_Ifc_Entry_Ptr->Designated_Router);
    printf("\nBackup_Designated_Router: %d",IPv4_Ifc_Entry_Ptr->Backup_Designated_Router);
    printf("\nCost: %d",IPv4_Ifc_Entry_Ptr->Cost);
    printf("\nPPP_Connection_Profile: %s",IPv4_Ifc_Entry_Ptr->PPP_Connection_Profile);
    /*Increment IPv4_Ifc_Entry_Ptr bytes in order to point to the next structure*/
    IPv4_Ifc_Entry_Ptr+=sizeof(Qtoo_IPv4_OSPF_Ifc_Entry);
}

char *IPv6_Ifc_Ptr = (char*)(OSPF_Receiver);

/*Cast of the Qtoo_IPv6_OSPF_Ifc_Entry_t structure is made to the IPv6_Ifc_Entry_Ptr */
/*pointer and the displacement will be Offset_To_IPv6_Ifc_Lst bytes, to aim at the */
/*Offset_To_IPv6_Ifc_Lst of the list in the structure */

Qtoo_IPv6_OSPF_Ifc_Entry_t *IPv6_Ifc_Entry_Ptr=(Qtoo_IPv6_OSPF_Ifc_Entry_t*)(IPv6_Ifc_Ptr+OSPF_Ptr-
>Offset_To_IPv6_Ifc_Lst);
for(unsigned long i=0;i < OSPF_Ptr->Number_Of_IPv6_Ifc_Lst_Ent;i++)
{
    printf("\nPrinting IPv6 Interface Entry %d",i);
    printf("\nIPv6_Address: %s",IPv6_Ifc_Entry_Ptr->IPv6_Address);
    printf("\nArea_ID: %s",IPv6_Ifc_Entry_Ptr->AreaID);
    printf("\nInterface_Type: %d",IPv6_Ifc_Entry_Ptr->Interface_Type);
    printf("\nDesignated_Router_Pri: %d",IPv6_Ifc_Entry_Ptr->Designated_Router_Pri);
    printf("\nTransmission_Dly: %d",IPv6_Ifc_Entry_Ptr->Transmission_Dly);
    printf("\nRetransmission_Dly: %d",IPv6_Ifc_Entry_Ptr->Retransmission_Dly);
    printf("\nHello_Interval: %d",IPv6_Ifc_Entry_Ptr->Hello_Interval);
    printf("\nInactive_Router_Interval: %d",IPv6_Ifc_Entry_Ptr->Inactive_Router_Interval);
    printf("\nPoll_Interval: %d",IPv6_Ifc_Entry_Ptr->Poll_Interval);
    printf("\nInterface_State: %d",IPv6_Ifc_Entry_Ptr->Interface_State);
    printf("\nDesignated_Router: %d",IPv6_Ifc_Entry_Ptr->Designated_Router);
    printf("\nBackup_Designated_Router: %d",IPv6_Ifc_Entry_Ptr->Backup_Designated_Router);
    printf("\nCost: %d",IPv6_Ifc_Entry_Ptr->Cost);
    /*Increment IPv6_Ifc_Entry_Ptr bytes in order to point to the next structure*/
    IPv6_Ifc_Entry_Ptr+=sizeof(Qtoo_IPv6_OSPF_Ifc_Entry);
}

char *IPv4_Neig_Ptr = (char*)(OSPF_Receiver);

/*Cast of the Qtoo_IPv4_OSPF_Neighbor_Entry_t structure is made to the IPv4_Ifc_Entry_Ptr */
/*pointer and the displacement will be Offset_To_IPv4_Neigh_Lst bytes, to aim at the */
/*Offset_To_IPv4_Neigh_Lst of the list in the structure */

Qtoo_IPv4_OSPF_Neighbor_Entry_t *IPv4_Neig_Entry_Ptr=(Qtoo_IPv4_OSPF_Neighbor_Entry_t*)(IPv4_Neig_Ptr+OSPF_Ptr-
>Offset_To_IPv4_Neigh_Lst);
for(unsigned long i=0;i < OSPF_Ptr->Number_Of_IPv4_Neigh_Lst_Ent;i++)
{
    printf("\nPrinting IPv4 Neighbor Entry %d",i);
    printf("\nIfc_IP_address: %s",IPv4_Neig_Entry_Ptr->Ifc_IP_address);
    printf("\nNeighbor_IP_Address: %s",IPv4_Neig_Entry_Ptr->Neighbor_IP_Address);
    printf("\nRouterID: %s",IPv4_Neig_Entry_Ptr->RouteID);
    printf("\nNeighbor_Options: %d",IPv4_Neig_Entry_Ptr->Neighbor_Options);
    printf("\nNeighbor_Priority: %d",IPv4_Neig_Entry_Ptr->Neighbor_Priority);
    printf("\nNeighbor_State: %d",IPv4_Neig_Entry_Ptr->Neighbor_State);
    printf("\nNeighbor_Events: %d",IPv4_Neig_Entry_Ptr->Neighbor_Events);
    /*Increment IPv4_Neig_Entry_Ptr bytes in order to point to the next structure*/
    IPv4_Neig_Entry_Ptr+=sizeof(Qtoo_IPv4_OSPF_Neighbor_Entry);
}

char *IPv6_Neig_Ptr = (char*)(OSPF_Receiver);

/*Cast of the Qtoo_IPv6_OSPF_Neighbor_Entry_t structure is made to the IPv6_Ifc_Entry_Ptr */
/*pointer and the displacement will be Offset_To_IPv6_Neigh_Lst bytes, to aim at the */
/*Offset_To_IPv6_Neigh_Lst of the list in the structure */

Qtoo_IPv6_OSPF_Neighbor_Entry_t *IPv6_Neig_Entry_Ptr=(Qtoo_IPv6_OSPF_Neighbor_Entry_t*)(IPv6_Neig_Ptr+OSPF_Ptr-
>Offset_To_IPv6_Neigh_Lst);
for(unsigned long i=0;i < OSPF_Ptr->Number_Of_IPv6_Neigh_Lst_Ent;i++)
{
    printf("\nPrinting IPv6 Neighbor Entry %d",i);
}

```

```

printf("\nIfc_IP_address: %s",IPv6_Neig_Entry_Ptr->Ifc_IP_address);
printf("\nNeighbor_IP_Address: %s",IPv6_Neig_Entry_Ptr->Neighbor_IP_Address);
printf("\nRouterID: %s",IPv6_Neig_Entry_Ptr->RouterID);
printf("\nNeighbor_Options: %d",IPv6_Neig_Entry_Ptr->Neighbor_Options);
printf("\nNeighbor_Priority: %d",IPv6_Neig_Entry_Ptr->Neighbor_Priority);
printf("\nNeighbor_State: %d",IPv6_Neig_Entry_Ptr->Neighbor_State);
printf("\nNeighbor_Events: %d",IPv6_Neig_Entry_Ptr->Neighbor_Events);
/*Increment IPv6_Neig_Entry_Ptr bytes in order to point to the next structure*/
IPv6_Neig_Entry_Ptr+=sizeof(Qtoo_IPv6_OSPF_Neighbor_Entry);
}

char *IPv4_Vtl_Ptr = (char*)(OSPF_Receiver);

/*Cast of the Qtoo_IPv4_OSPF_Vtl_Entry_t structure is made to the IPv4_Ifc_Entry_Ptr */
/*pointer and the displacement will be Offset_To_IPv4_Vtl_Lst bytes, to aim at the */
/*Offset_To_IPv4_Vtl_Lst of the list in the structure */

Qtoo_IPv4_OSPF_Vtl_Link_Entry_t *IPv4_Vtl_Entry_Ptr=(Qtoo_IPv4_OSPF_Vtl_Link_Entry_t*)(IPv4_Vtl_Ptr+OSPF_Ptr-
>Offset_To_IPv4_Vtl_Lst);
for(unsigned long i=0;i < OSPF_Ptr->Number_Of_IPv4_Vtl_Lst_Ent;i++)
{
printf("\nPrinting IPv4 Virtual Link Entry %d",i);
printf("\nTransist_Area: %s",IPv4_Vtl_Entry_Ptr->Transist_Area);
printf("\nRouterID: %s",IPv4_Vtl_Entry_Ptr->RouterID);
printf("\nVtl_Link_Transit_Dly: %s",IPv4_Vtl_Entry_Ptr->Vtl_Link_Transit_Dly);
printf("\nVtl_Link_Retransmission_Dly: %s",IPv4_Vtl_Entry_Ptr->Vtl_Link_Retransmission_Dly);
printf("\nVtl_Link_Hello_Interval: %s",IPv4_Vtl_Entry_Ptr->Vtl_Link_Hello_Interval);
printf("\nVtl_Link_State: %s",IPv4_Vtl_Entry_Ptr->Vtl_Link_State);
/*Increment IPv4_Vtl_Entry_Ptr bytes in order to point to the next structure*/
IPv4_Vtl_Entry_Ptr+=sizeof(Qtoo_IPv4_OSPF_Vtl_Link_Entry);
}

char *IPv6_Vtl_Ptr = (char*)(OSPF_Receiver);

/*Cast of the Qtoo_IPv6_OSPF_Vtl_Entry_t structure is made to the IPv6_Ifc_Entry_Ptr */
/*pointer and the displacement will be Offset_To_IPv6_Vtl_Lst bytes, to aim at the */
/*Offset_To_IPv6_Vtl_Lst of the list in the structure */

Qtoo_IPv6_OSPF_Vtl_Link_Entry_t *IPv6_Vtl_Entry_Ptr=(Qtoo_IPv6_OSPF_Vtl_Link_Entry_t*)(IPv6_Vtl_Ptr+OSPF_Ptr-
>Offset_To_IPv6_Vtl_Lst);
for(unsigned long i=0;i < OSPF_Ptr->Number_Of_IPv6_Vtl_Lst_Ent;i++)
{
printf("\nPrinting IPv6 Virtual Link Entry %d",i);
printf("\nTransist_Area: %s",IPv6_Vtl_Entry_Ptr->Transist_Area);
printf("\nRouterID: %s",IPv6_Vtl_Entry_Ptr->RouterID);
printf("\nVtl_Link_Transit_Dly: %s",IPv6_Vtl_Entry_Ptr->Vtl_Link_Transit_Dly);
printf("\nVtl_Link_Retransmission_Dly: %s",IPv6_Vtl_Entry_Ptr->Vtl_Link_Retransmission_Dly);
printf("\nVtl_Link_Hello_Interval: %s",IPv6_Vtl_Entry_Ptr->Vtl_Link_Hello_Interval);
printf("\nVtl_Link_State: %s",IPv6_Vtl_Entry_Ptr->Vtl_Link_State);
/*Increment IPv6_Vtl_Entry_Ptr bytes in order to point to the next structure*/
IPv6_Vtl_Entry_Ptr+=sizeof(Qtoo_IPv6_OSPF_Vtl_Link_Entry);
}

char *IPv4_LSA_Ptr = (char*)(OSPF_Receiver);

/*Cast of the Qtoo_IPv4_OSPF_LSA_Entry_t structure is made to the IPv4_Ifc_Entry_Ptr */
/*pointer and the displacement will be Offset_To_IPv4_LSA_Lst bytes, to aim at the */
/*Offset_To_IPv4_LSA_Lst of the list in the structure */

Qtoo_IPv4_OSPF_LSA_Entry_t *IPv4_LSA_Entry_Ptr=(Qtoo_IPv4_OSPF_LSA_Entry_t*)(IPv4_LSA_Ptr+OSPF_Ptr-
>Offset_to_IPv4_LSA_Lst);
for(unsigned long i=0;i < OSPF_Ptr->Number_Of_IPv4_LSA_Lst_Ent;i++)
{
printf("\nPrinting IPv4 LSA Entry %d",i);
printf("\nLSA_Area_ID: %s",IPv4_LSA_Entry_Ptr->LSA_Area_ID);
printf("\nLSA_Type: %d",IPv4_LSA_Entry_Ptr->LSA_Type);
printf("\nLSA_State_ID: %s",IPv4_LSA_Entry_Ptr->LSA_State_ID);
printf("\nLSA_Router_ID: %s",IPv4_LSA_Entry_Ptr->LSA_Router_ID);
printf("\nLSA_Sequence: %d",IPv4_LSA_Entry_Ptr->LSA_Sequence);
printf("\nLSA_Age: %d",IPv4_LSA_Entry_Ptr->LSA_Age);
printf("\nLSA_Checksum: %d",IPv4_LSA_Entry_Ptr->LSA_Checksum);
/*Increment IPv4_LSA_Entry_Ptr bytes in order to point to the next structure*/
IPv4_LSA_Entry_Ptr+=sizeof(Qtoo_IPv4_OSPF_LSA_Entry_t);
}

char *IPv6_LSA_Ptr = (char*)(OSPF_Receiver);

/*Cast of the Qtoo_IPv6_OSPF_LSA_Entry_t structure is made to the IPv4_Ifc_Entry_Ptr */
/*pointer and the displacement will be Offset_To_IPv6_LSA_Lst bytes, to aim at the */
/*Offset_To_IPv6_LSA_Lst of the list in the structure */

Qtoo_IPv6_OSPF_LSA_Entry_t *IPv6_LSA_Entry_Ptr=(Qtoo_IPv6_OSPF_LSA_Entry_t*)(IPv6_LSA_Ptr+OSPF_Ptr-
>Offset_to_IPv6_LSA_Lst);
for(unsigned long i=0;i < OSPF_Ptr->Number_Of_IPv6_LSA_Lst_Ent;i++)
{
printf("\nPrinting IPv6 LSA Entry %d",i);
printf("\nLSA_Area_ID: %s",IPv6_LSA_Entry_Ptr->LSA_Area_ID);
printf("\nLSA_Type: %d",IPv6_LSA_Entry_Ptr->LSA_Type);
printf("\nLSA_State_ID: %s",IPv6_LSA_Entry_Ptr->LSA_State_ID);
printf("\nLSA_Router_ID: %s",IPv6_LSA_Entry_Ptr->LSA_Router_ID);
printf("\nLSA_Sequence: %d",IPv6_LSA_Entry_Ptr->LSA_Sequence);
printf("\nLSA_Age: %d",IPv6_LSA_Entry_Ptr->LSA_Age);
printf("\nLSA_Checksum: %d",IPv6_LSA_Entry_Ptr->LSA_Checksum);
}

```

```
/*Increment IPv6_LSA_Entry_Ptr bytes in order to point to the next structure*/  
IPv6_LSA_Entry_Ptr+=sizeof(Qtoo_IPv6_OSPF_LSA_Entry);  
}  
}
```

Related concepts

Open Shortest Path First API and commands

Use control language (CL) commands to manage OSPF attributes, interfaces, areas, ranges, links, statistics, state information, and to log OSPF activity. Use the Retrieve OSPF State Information API to retrieve OSPF statistics.

Code license and disclaimer information

IBM® grants you a nonexclusive copyright license to use all programming code examples from which you can generate similar function tailored to your own specific needs.

SUBJECT TO ANY STATUTORY WARRANTIES WHICH CANNOT BE EXCLUDED, IBM, ITS PROGRAM DEVELOPERS AND SUPPLIERS MAKE NO WARRANTIES OR CONDITIONS EITHER EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OR CONDITIONS OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE, AND NON-INFRINGEMENT, REGARDING THE PROGRAM OR TECHNICAL SUPPORT, IF ANY.

UNDER NO CIRCUMSTANCES IS IBM, ITS PROGRAM DEVELOPERS OR SUPPLIERS LIABLE FOR ANY OF THE FOLLOWING, EVEN IF INFORMED OF THEIR POSSIBILITY:

1. LOSS OF, OR DAMAGE TO, DATA;
2. DIRECT, SPECIAL, INCIDENTAL, OR INDIRECT DAMAGES, OR FOR ANY ECONOMIC CONSEQUENTIAL DAMAGES; OR
3. LOST PROFITS, BUSINESS, REVENUE, GOODWILL, OR ANTICIPATED SAVINGS.

SOME JURISDICTIONS DO NOT ALLOW THE EXCLUSION OR LIMITATION OF DIRECT, INCIDENTAL, OR CONSEQUENTIAL DAMAGES, SO SOME OR ALL OF THE ABOVE LIMITATIONS OR EXCLUSIONS MAY NOT APPLY TO YOU.

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:

Intellectual Property Licensing
Legal and Intellectual Property Law
IBM Japan Ltd.
1623-14, Shimotsuruma, Yamato-shi
Kanagawa 242-8502 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:

IBM Corporation
Software Interoperability Coordinator, Department YBWA
3605 Highway 52 N
Rochester, MN 55901
U.S.A.

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 document and all licensed material available for it are provided by IBM under terms of the IBM Customer Agreement, IBM International Program License Agreement 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.

All statements regarding IBM's future direction or intent are subject to change or withdrawal without notice, and represent goals and objectives only.

All IBM prices shown are IBM's suggested retail prices, are current and are subject to change without notice. Dealer prices may vary.

This information is for planning purposes only. The information herein is subject to change before the products described become available.

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. The sample programs are provided "AS IS", without warranty of any kind. IBM shall not be liable for any damages arising out of your use of the sample programs.

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

© (your company name) (year). Portions of this code are derived from IBM Corp. Sample Programs.

© Copyright IBM Corp. _enter the year or years_.

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

Programming interface information

This publication documents intended Programming Interfaces that allow the customer to write programs to obtain the services of IBM i.

Trademarks

IBM, the IBM logo, and ibm.com are trademarks or registered trademarks of International Business Machines Corp., registered in many jurisdictions worldwide. Other product and service names might be

trademarks of IBM or other companies. A current list of IBM trademarks is available on the Web at "[Copyright and trademark information](http://www.ibm.com/legal/copytrade.shtml)" at www.ibm.com/legal/copytrade.shtml.

Adobe, the Adobe logo, PostScript, and the PostScript logo are either registered trademarks or trademarks of Adobe Systems Incorporated in the United States, and/or other countries.

IT Infrastructure Library is a registered trademark of the Central Computer and Telecommunications Agency which is now part of the Office of Government Commerce.

Intel, Intel logo, Intel Inside, Intel Inside logo, Intel Centrino, Intel Centrino logo, Celeron, Intel Xeon, Intel SpeedStep, Itanium, and Pentium are trademarks or registered trademarks of Intel Corporation or its subsidiaries in the United States and other countries.

Linux® is a registered trademark of Linus Torvalds in the United States, other countries, or both.

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

ITIL is a registered trademark, and a registered community trademark of the Office of Government Commerce, and is registered in the U.S. Patent and Trademark Office.

UNIX is a registered trademark of The Open Group in the United States and other countries.

Cell Broadband Engine is a trademark of Sony Computer Entertainment, Inc. in the United States, other countries, or both and is used under license therefrom.

Java™ and all Java-based trademarks and logos are trademarks of Oracle, Inc. in the United States, other countries, or both.

Other product and service names might be trademarks of IBM or other companies.

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.



Product Number: 5770-SS1