

DoD Internet Protocol Static Routing

Internal Design Specification

December 18, 1985

AUTHOR: _____
John R. Lyman III

TDRB APPROVAL: _____

AD&C APPROVAL: _____

DISCLAIMER: This document is an internal working paper only. It is unapproved, subject to change, and does not necessarily represent any official intent on the part of Control Data Corporation.

ACN LIBRARY
 ARH207 M.J. AMUNDSON
 ARH207 P.C. HAHN
 ARH207 A.A. WEBER
 ARH207 R.E. ERICKSON
 ARH207 K.D. DEHNERT
 ARH207 G.S. JUNCKER
 ARH207 T.C. MCGEE
 ARH207 M.E. MADDEN
 ARH213 J.D. KNOJEL
 ARH213 C.J. DUFFY
 ARH242 R.A. JAPS
 ARH254 J.E. ESLER
 ARH293 N.L. REDDY
 AHS279 J.R. HILDEBRAND
 PLY011 R.L. CLARK
 STAOPS M.C. STEELE

P.A. SHIREMAN
 B.T. ZEMLIN
 J.P. YOUNG
 R.R. RUNDQUIST
 R.E. TATE
 H.G. COVERSTON
 D.A. NETLAND
 J.D. REED
 R.D. SWAN
 B.S. SEKHON
 K.G. VIGGERS
 D.K. TAYLOR
 J.A. WACHTER
 S.M. KEEFER
 J.A. DAYKIN
 E.E. NELSON
 L.L. LUCAS
 R.L. SUNDBERG

SVL107
 ARH207
 ARH207
 ARH207
 ARH207
 ARH207
 ARH207
 ARH207
 ARH207
 ARH207
 ARH207
 ARH207
 ARH207
 ARH207
 ARH207
 ARH207
 ARH207
 ARH207
 ARH207
 ARH207

CONTROL DATA CORPORATION
 DOCUMENT CONTROL FORM

DOCUMENT TYPE: **IDS** LOG ID: **ARH7141**

TITLE: **DOD Internet Protocol Static Routing IDS**

ABSTRACT: **This document describes the internal design of the CDCNET implementation of static routing for DOD's Internet Protocol**

PRODUCT AFFECTED: **CDCNET, DCNS**

AUTHOR: **J. R. Lyman**

MAIL STATION: **ARH207** EXTENSION: **X3367**

PROJECT: **ACSD Development**

SECTION:

RSM or PSR #: _____ SIP Number(s): _____

PUB #: _____

PRM #: _____ Redesign* Reimplementation*

| Name | Approval Initials | Date |
|---|-------------------|---------------|
| Internal Reviewer* | | |
| Project Leader | | |
| Unit Manager: T. C. McGee | T.C.M. | 1-8-86 |
| Section Manager | | |
| Design Team Referee: B.S. Sekhon | BSS | 1-8-86 |

DESIGN TEAM: _____ DEFAULT CYCLE: **0** WORKING DAYS

Special Distribution: **M. R. Sanders SVL069B**

Referee's Distribution Codes: **ACCG-G**

| NUMBER | DATE | TYPE | SUBMITTED BY | DEFAULT APPROVAL DATE | CURRENT STATUS | | | WITH-DRAWN |
|--------|--------|------|--------------|-----------------------|----------------|------|----------|------------|
| | | | | | REVIEW | HOLD | APPROVED | |
| 1 | 1/9/86 | Orig | J. R. Lyman | NONE | | | A | |
| | | | | | | | | |
| | | | | | | | | |
| | | | | | | | | |
| | | | | | | | | |

85/12/19

PROPRIETARY NOTICE

The technical content set forth in this document is the property of Control Data and is not to be disseminated, distributed or otherwise conveyed to third parties without the express written permission of Control Data.

This document is to be used for planning purposes only. It does not include any explicit or implied commitments for any specific release dates or feature content. These matters are currently under active development, and as such are subject to revision and replanning as circumstances warrant.

85/12/19

| RECORD OF REVISION | | | |
|--------------------|---------------------------|--------|----------|
| Revision | Description | Author | Date |
| 01 | Draft Version | JRL3 | 11/21/85 |
| 02 | Revised for TDRB comments | JRL3 | 12/18/85 |
| | | | |
| | | | |
| | | | |
| | | | |
| | | | |
| | | | |
| | | | |
| | | | |
| | | | |
| | | | |
| | | | |
| | | | |
| | | | |

85/12/19

1.0 INTRODUCTION

1.0 INTRODUCTION

This document describes the internal design of the DoD Internet Protocol Static Routing (IPSR) module. The IPSR module contains the routing tables that are used by all other DoD protocols. Routines are provided to maintain the table of directly connected hosts and the table of reachable networks. Command processors are provided to maintain both of these tables. The major purpose of this module is fulfilled by a routine provided to determine the next hop for a given datagram.

The IPSR routines run under the task of the caller. The module is basically a set of routines that hide the routing table. The routines which require memory allocation will return appropriate error messages.

85/12/19

2.0 REFERENCES

2.0 REFERENCES

The following manuals contain material that either defines the operations of the TCP module and the modules it interfaces to, or provides additional insight into the use of the TCP module.

- | | | | |
|-----|--------------|-----|---------------------------------------|
| [1] | RFC-791 | SRI | DoD Internet Protocol |
| [2] | RFC-792 | SRI | DoD Internet Control Message Protocol |
| [3] | MIL-STD-1777 | DoD | DoD Internet Protocol Standard |
| [4] | ARH6265 | CDC | DoD Internet Protocol ERS |
| [5] | ARH7118 | CDC | Dod IP Static Routing ERS |
| [6] | ARH7016 | CDC | DoD Internet Protocol IDS |

85/12/19

3.0 ENVIRONMENT

3.0 ENVIRONMENT

3.1 HARDWARE

The IPSR module has no special hardware requirements. The IPSR module is part of the CDCNET software and will run on a 68000 based Device Interface (DI). The module will be written in the CYBIL language, and will be compiled and bound using SES tools on a CYBER mainframe.

3.2 SOFTWARE

The IPSR module depends on a number of other software components in order to function in the DI. The following sections list these components and itemize the services of each component that the IPSR module uses.

3.2.1 INTERNET PROTOCOL MODULE

The IP module provides an indication for each ICMP redirect datagram that it receives. This interface allows an IPSR module in a host which contains no other routing protocol to keep its network table dynamically updated.

3.2.2 3A INTRANET MODULE

The 3A Intranet module provides a basic point-to-point data transfer service between two systems connected to a common network solution. The IPSR module will open a SAP with the 3A module in order to receive status information about the IP networks that the 3A module supports. The IPSR module will expect to receive the following information from the 3A module for each such IP network

85/12/19

3.0 ENVIRONMENT

3.2.2 3A INTRANET MODULE

1. The IP address of the network connection.
2. The 3A System ID.
3. The 3A network ID.
4. The maximum packet size.
5. And the current status of the network.

3.2.3 STATUS COMMAND PROCESSOR

The IPSR module will provide a report procedure that the status command processor can call to display the contents of the routing tables.

3.2.4 EXECUTIVE COMMON ROUTINES

The IPSR module will use a number of the common subroutines that are provided as part of the executive. The following services will be expected from the executive routines.

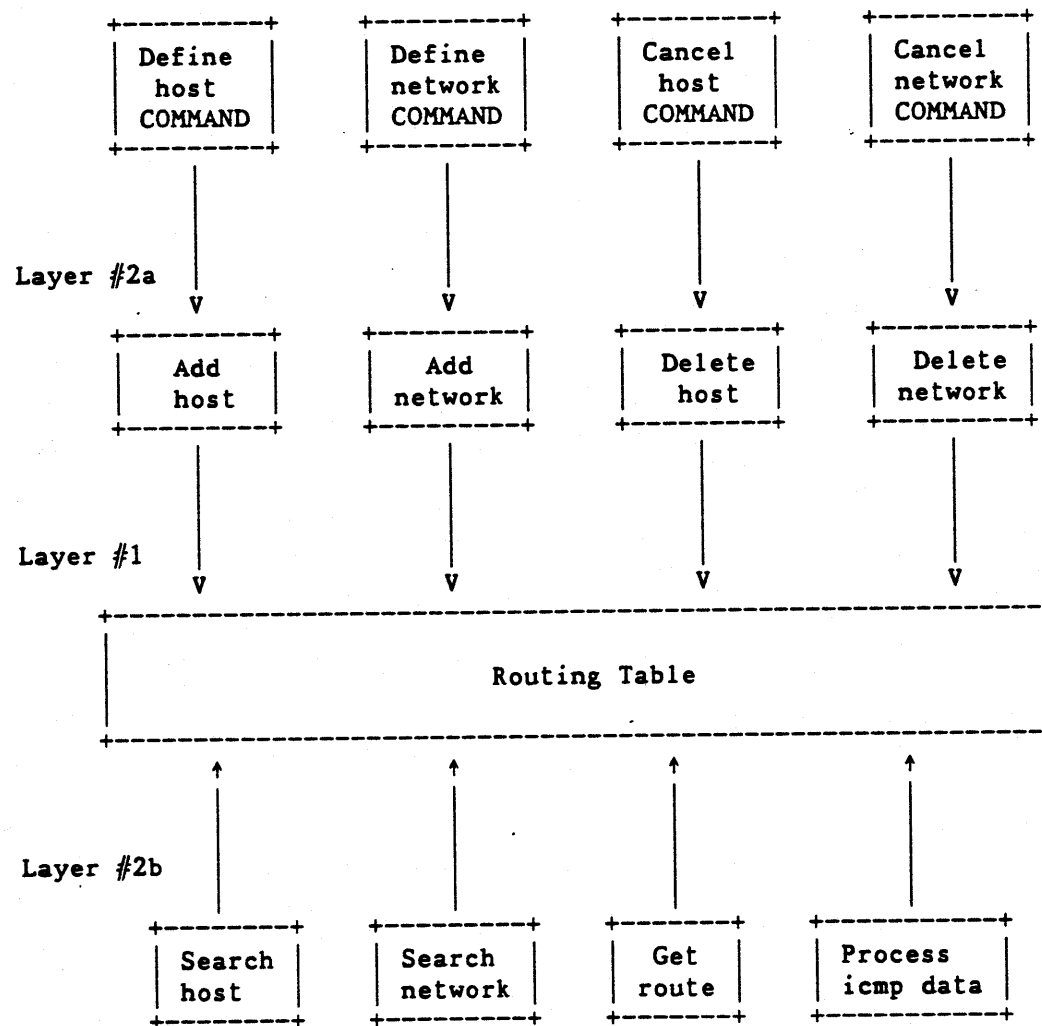
1. Buffer management.
2. Memory allocation.
3. Memory deallocation.
4. Binary Tree Table Management.
5. Timer management.

4.0 DESIGN OVERVIEW

4.0 DESIGN OVERVIEW

The IPSR module is composed of three layers of software. The following diagram illustrates the components of each layer, and the relationships between the layers.

Layer #3



85/12/19

4.0 DESIGN OVERVIEW

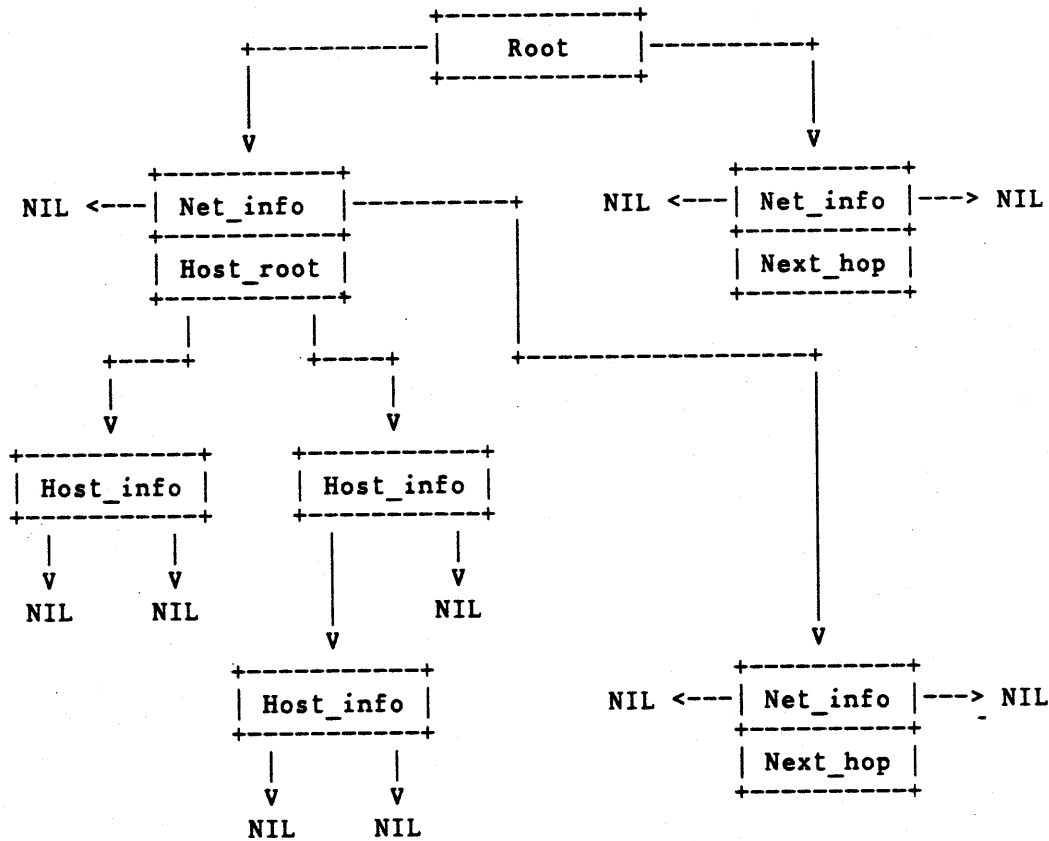
The first layer is composed of the routing table itself. This table will contain all information necessary to determine the next stop in a datagrams route. The table will be sorted so that the time needed to determine that next stop will be minimized. In fact, the time needed to modify the table will be considered unimportant. It is assumed that a Gateway Protocol module will contain tables which it will use in addition to the routing table.

The second layer contains the routines that maintain the routing table. These routines allow the host/network table to be updated directly and are used to process ICMP datagrams. This layer also contains the routine which IP calls to determine the next stop for a given datagram. All of these routines are intimately familiar with the structure of the routing tables.

The final layer contains four command processors. These command processors are intended to allow the network operator to maintain the routing table manually. This will be necessary as long as there is no dynamic routing protocol (Gateway Protocol) available. These command processors will make calls to the layer two routines, they will therefore, do very little actual processing.

4.0 DESIGN OVERVIEW

The routing table is composed of a nested binary tree. The main tree contains a node for each network that the IPSR module knows about. Each network node will either be locally or remotely connected. The nodes for remotely connected networks will indicate the next hop to that network. The nodes for locally connected networks will contain a binary tree of all the hosts known to reside on that network. The following diagram specifies a possible structure of the table.



85/12/19

4.0 DESIGN OVERVIEW

4.1 FUNCTIONAL STRUCTURE

4.1 FUNCTIONAL STRUCTURE

4.1.1 SERVICE_ROUTINES

The routines described in this section are provided for the use of any module that needs to update the routing tables. The use of the routines in this section allows a user to restrict his knowledge of the internal structure of the routing tables to the format of an individual entry. The search routines allow the user to access a specific entry in a routing table without knowing how the entries are arranged in the table.

4.1.1.1 Add host

This routine is used to add the information about a directly connected host into the routing table. Each IP host on a directly connected IP network must be entered into the routing table with this routine. All hosts in a connected CDC Catenet to whom a gateway provides direct service, are considered to be directly connected to that gateway and must be entered into its routing tables with this routine. The format of the CYBIL interface is as follows:

Call format

```
PROCEDURE ipsr_add_host (  
  host_address : ip_address;  
  host_type    : ipsr_host_type;  
  egp_active   : BOOLEAN;  
  igp_active   : BOOLEAN;  
  3a_network   : net_id_type;  
  3a_system    : sys_id_type;  
  VAR status   : ipsr_status_type);
```

host_address In This is the IP address of the
 host being defined.

host_type In This is the type of host being
 defined.

CONTROL DATA PRIVATE

85/12/19

4.0 DESIGN OVERVIEW4.1.1.1 Add_host

| | | |
|------------|-----|---|
| egp_active | In | This flag is set if an external gateway protocol is active in this host. |
| igp_active | In | This flag is set if an internal gateway protocol is active in this host. |
| 3a_network | In | This is the 3A net_id of the network solution that the host is connected to. |
| 3a_system | In | This is the 3A system_id of the host. |
| status | Out | This is the status of the request. The following values may be returned: iprs_successful iprs_host_exists iprs_invalid_address iprs_invalid_type iprs_unknown_network iprs_insufficient_resources |

85/12/19

4.0 DESIGN OVERVIEW4.1.1.1 Add_host

Global data accessed

1. The network/host table will be accessed, and may be updated.

General Algorithm

```
BEGIN
  IF (INVALID parameter) THEN
    RETURN(error_message);
  ELSE
    CALL ipsr_search_host_table (for the specified host);
    IF found THEN
      RETURN(ipsr_host_exists);
    ELSE
      Search for the network;
      IF (NOT found) OR (NOT directly connected) THEN
        RETURN(ipsr_unknown_network);
      ELSE
        Add the host to the networks host table;
        IF addition successful THEN
          RETURN(ipsr_successful);
        ELSE
          RETURN(ipsr_insufficient_resources);
        IFEND;
      IFEND;
    IFEND;
  IFEND;
END
```

85/12/19

4.0 DESIGN OVERVIEW4.1.1.2 Add_network
-----4.1.1.2 Add network

This routine is used to add information about a reachable network into the routing table. This routine will be used at first by the command processor which adds networks. It may also be used by modules that provide dynamic routing services. The format of the CYBIL interface is as follows:

Call format

```
PROCEDURE ipsr_add_network (
  network   : 0..OFFFFFF(16);
  gateway   : ip_address;
  hop_count : INTEGER;
  owner     : ipsr_net_owners;
  VAR status : ipsr_status_type);
```

| | | |
|-----------|-----|---|
| network | In | This is the network number of the network that is being added to the routing table. |
| gateway | In | This is the IP address of the gateway that datagrams destined for this network should be sent to. |
| hop_count | In | This is the number of gateways that the datagrams must pass through to reach this network. |
| owner | In | This is the owner of the table entry. Dynamic routing modules may use this value to insure that they only delete entries that they added. |
| status | Out | This is the status of the request. The following values may be returned: ipsr_successful ipsr_invalid_owner ipsr_network_exists ipsr_unknown_gateway ipsr_insufficient_resources |

85/12/19

4.0 DESIGN OVERVIEW**4.1.1.2 Add_network**

Global data accessed

1. The network/host table will be accessed, and may be updated.

General Algorithm

```
BEGIN
  IF (owner INVALID) THEN
    RETURN(ipsr_invalid_owner);
  ELSE
    Search for the specified network;
    IF found THEN
      RETURN(ipsr_network_exists);
    ELSE
      CALL ipsr_search_host_table (for the gateway);
      IF NOT found THEN
        RETURN(ipsr_unknown_gateway);
      ELSE
        Add the network to the table;
        IF addition successful THEN
          RETURN(ipsr_successful);
        ELSE
          RETURN(ipsr_insufficient_resources);
        IFEND;
      IFEND;
    IFEND;
  IFEND;
END
```


85/12/19

4.0 DESIGN OVERVIEW**4.1.1.3 Delete_host**

4.1.1.3 Delete host

This routine is used to delete a host from the routing table. When an existing entry needs to be changed, the current entry must be deleted and then the new entry added. The format of the CYBIL interface is as follows:

Call format

```
PROCEDURE ipsr_delete_host (  
    host_address : ip_address;  
    VAR status   : ipsr_status_type);
```

| | | |
|--------------|-----|--|
| host_address | In | This is the IP address of the host. |
| status | Out | This is the status of the request. The following values may be returned: ipsr_successful ipsr_unknown_host ipsr_invalid_address |

Global data accessed

1. The network/host table will be accessed, and may be updated.

4.0 DESIGN OVERVIEW
4.1.1.3 Delete_host

General Algorithm

```
BEGIN
  IF (host_address INVALID) THEN
    RETURN(ipsr_invalid_address);
  ELSE
    CALL ipsr_search_host_table (for the specified host);
    IF NOT found THEN
      RETURN(ipsr_unknown_host);
    ELSE
      Delete the host entry;
      IF (host_type=ip_gw) OR (host_type=cdc_gw) THEN
        Update default gateway if needed;
        Change networks using this gateway to default;
      IFEND;
      RETURN(ipsr_successful);
    IFEND;
  IFEND;
END
```

85/12/19

4.0 DESIGN OVERVIEW4.1.1.4 Delete_network

4.1.1.4 Delete network

This routine is used to delete information about a reachable network from the routing table. This routine will be used at first by the command processor which removes networks. It may also be used by modules that provide dynamic routing services. The format of the CYBIL interface is as follows:

Call format

```
PROCEDURE ipsr_delete_network (
  network      : 0..OFFFFFF(16);
  owner        : ipsr_net_owners;
  VAR status   : ipsr_status_type);
```

| | | |
|---------|-----|---|
| network | In | This is the network number of the network that is being removed from the routing table. |
| owner | In | This is the owner of the table entry. Dynamic routing modules may use this value to insure that they only delete entries that they added. |
| status | Out | This is the status of the request. The following values may be returned: ipsr_successful ipsr_invalid_owner ipsr_unknown_network |

Global data accessed

1. The network/host table will be accessed, and may be updated.

4.0 DESIGN OVERVIEW
4.1.1.4 Delete_network

General Algorithm

```
BEGIN
  IF (owner INVALID) THEN
    RETURN(ipsr_invalid_owner);
  ELSE
    Search for the specified network;
    IF NOT found THEN
      RETURN(ipsr_unknown_network);
    ELSE
      IF (network_type=direct) THEN
        Delete the host table;
      IFEND;
      Delete the network entry;
      RETURN(ipsr_successful);
    IFEND;
  IFEND;
END
```

85/12/19

4.0 DESIGN OVERVIEW4.1.1.5 Get_route
-----4.1.1.5 Get route

This routine is called by the IP module when a datagram is received. The IPSR module uses the destination address, the source address, and the ip routing options to determine the best place for the datagram to go next. The source address is optional and the IPSR module will determine it and return it to the IP module if not specified. The 3A network id and the 3A system id of the local destination are returned to the IP module along with the type of destination. This routine will also process the IP routing options, if there are any, and updates them according to the IP protocol. The format of the CYBIL interface is as follows:

Call format

```

PROCEDURE ipsr_get_route (
    from_network : BOOLEAN;
    header       : ip_header;
    VAR source   : ip_address;
    destination  : ip_address;
    VAR options  : ip_option_record;
    VAR host_info : ipsr_host_rec;
    VAR max_data_size : INTEGER;
    VAR status   : ipsr_status_type);

```

WHERE DATAGRAM CAME FROM, IF NOT HERE.

3A_Sys only!

| | | |
|--------------|-----|--|
| from_network | In | This flag should be TRUE if the datagram that is being routed was received from a network. |
| header | In | This is the header of the IP datagram. |
| source | I/O | This is the IP address of the source, this should be the address of the DI on a directly connected network. This parameter is optional and will be chosen by the IPSR module if not specified. |
| destination | In | This is the destination of the IP datagram. |

4.0 DESIGN OVERVIEW

4.1.1.5 Get_route

| | | |
|----------------------|-----|---|
| options | I/O | This is an array which contains the IP routing options. |
| host_info | Out | This is a record which contains the routing information that the IP module requires. |
| max_data_size | Out | This is the maximum number of bytes that a datagram being sent to the indicated destination can contain. |
| status | Out | This is the status of the request. The following values may be returned: ipr_successful ipr_unable_to_route ipr_invalid_option |

Host-type
 net-ID
 sys-ID

Global data accessed

1. The network/host table will be searched.

85/12/19

 4.0 DESIGN OVERVIEW
 4.1.1.5 Get_route

General Algorithm

```

BEGIN
  IF parameter error THEN
    RETURN(appropriate message);
  ELSE
    Process the options in the header;
    IF routing options THEN
      Determine the next destination;
    IFEND;
    Search for the destination network;
    IF network found THEN
      IF network_type=direct THEN
        Search the network's host table;
        IF host not found THEN
          RETURN(ipshr_unable_to_route);
        IFEND
      ELSE
        Call ipshr_search_host_table(gateway);
        IF gateway not found THEN
          Log internal error;
          Use the default gateway;
        IFEND;
      IFEND;
      Copy info. from the network and host entries;
      IF source not specified THEN
        Find our local address on the dest. network;
        Record the local address as the source;
      IFEND;
      RETURN(ipshr_successful);
    ELSE
      IF NOT from_network THEN
        CALL ipshr_add_network(dest.net,default_gateway);
        Copy info. from the network and host entries;
        IF source not specified THEN
          Find our local address on the dest. network;
          Record the local address as the source;
        IFEND;
        RETURN(ipshr_successful);
      ELSE
        RETURN(ipshr_unable_to_route);
      IFEND;
    IFEND;
  END

```

Are these the same routine?

SEE MEMO 608 → MRS, JUL 3 2/27/86

RE. FEELS IN NET ENTRY SAYS: SEND IT OUT ON HOST .00 IF NET WILL FIND RIGHT HOST.

net host

85/12/19

4.0 DESIGN OVERVIEW**4.1.1.6 Process_icmp_data**

4.1.1.6 Process icmp data

This routine is called by the IP module to indicate that it has received an ICMP datagram. This is the only dynamic routing information that a standard DoD host will receive. The format of the CYBIL interface is as follows:

Call format

```
PROCEDURE ipsr_process_icmp_data (  
    error_type : ipsr_icmp_ind;  
    source      : ip_address;  
    destination : ip_address;  
    new_gateway : ip_address);
```

| | | |
|--------------------|-----------|---|
| error_type | In | This is the type of ICMP datagram that is being reported. |
| source | In | This is the source address from the datagram that the ICMP datagram pertains to. |
| destination | In | This is the destination address from the datagram that the ICMP datagram pertains to. |
| new_gateway | In | This is the IP address of the gateway that the datagram should have been sent to. |

Global data accessed

1. The network/host table will be updated.

85/12/19

4.0 DESIGN OVERVIEW
4.1.1.6 Process_icmp_data

General Algorithm

```
BEGIN
  Search for the specified network;
  IF network exists THEN
    CASE error_type OF
      =ipsr_end_source_quench=
        IF network.status=net_congested THEN
          network.status := net_up;
          network.timeouts := 0;
        IFEND;
      =ipsr_net_unreachable=
        network.hop_count := unreachable_hop;
        network.status := unknown;
      =ipsr_redirect=
        CALL ipsr_search_host_table(new_gateway);
        IF gateway found THEN
          network.hop_count := unknown_hop;
          network.timeouts := 0;
          next_hop := new_gateway_index;
        IFEND;
      =ipsr_source_quench=
        IF network.status=net_up THEN
          network.status := net_congested;
        IFEND;
      =ipsr_time_exceeded=
        IF (net.status=net_up) OR (net.status=unknown) THEN
          network.timeouts := network.timeouts + 1;
          IF network.timeouts>=timeout_threshold THEN
            network.hop_count := unreachable_hop;
            network.status := unknown;
          IFEND;
        IFEND;
    CASEEND;
  IFEND;
END
```

85/12/19

4.0 DESIGN OVERVIEW**4.1.1.7 Search_host_table**

4.1.1.7 Search host table

This routine will search the network/host table; the host_address will be used as the key. If an entry is found then a pointer to the host information record will be returned, else the pointer will be NIL.

Call format

```
PROCEDURE ipsr_search_host_table (  
    host_address : ip_address;  
    VAR host_entry : ^ipsr_host_info);
```

host_address In This is the address of the host
 that we are searching for.

host_entry Out This is a pointer to the
 information about the host if its
 found in the table. If the host is
 not found then this pointer is
 returned as NIL.

4.0 DESIGN OVERVIEW

4.1.1.7 Search_host_table

Global data accessed

1. The network/host table will be accessed.

General Algorithm

```
BEGIN
  Search for the specified network;
  IF (not found) OR (network_type=remote) THEN
    host_entry := NIL
  ELSE
    Search the networks host table;
    IF not found THEN
      host_entry := NIL
    ELSE
      host_entry points to the host entry;
    IFEND;
  IFEND;
END
```

85/12/19

4.0 DESIGN OVERVIEW**4.1.1.8 Search_network_table**

4.1.1.8 Search network table

This routine will search the network table. The network number will be used as the key. If an entry is found then a pointer to the network information record will be returned, else the pointer will be returned NIL.

Call format

```
PROCEDURE ipsr_search_network_table (  
  network_number : 0..FFFFFF;  
  VAR network_entry : ^ipsr_network_info);
```

network_number In This is the network number of
the network that we are searching
for.

network_entry Out This is a pointer to the
information record of the network
being searched for, if the network
is not found then the pointer will
be returned NIL.

Global data accessed

1. The network/host table will be accessed.

4.0 DESIGN OVERVIEW
4.1.1.8 Search_network_table

General Algorithm

```
BEGIN
  Search for the specified network;
  IF the network is found THEN
    network_entry points to the networks entry;
  ELSE
    network_entry := NIL;
  IFEND;
END
```

85/12/19

4.0 DESIGN OVERVIEW

4.1.2 COMMAND_PROCESSORS

4.1.2 COMMAND_PROCESSORS

The IPSR module provides four command processors. These commands allow the network operator to manually update the routing tables. In the initial releases these commands will be the only way to keep the routing tables up to date. In later releases, after dynamic routing capability is added, they will hopefully be needed very little.

4.1.2.1 Define ipsr host

This routine is called by the Command ME command processor interface task to process the Define_IPSR_host command. This command allows the network operator to add a host/gateway to the routing tables. For a definition of the procedure parameters, see the Command ME ERS.

Call format

```
PROCEDURE cmd_define_ipsr_host (  
    parameter_list : ost$string;  
    VAR pvt        : clt$parameter_value_table;  
    VAR c_code     : condition_code;  
    VAR status     : clt$status);
```

Global data accessed

1. The IPSR routine table will be updated.

4.0 DESIGN OVERVIEW
4.1.2.1 Define_ipsr_host

General Algorithm

```
BEGIN
  Process the parameters;
  IF all parameters valid THEN
    CALL ipsr_add_host;
    RETURN(appropriate message);
  ELSE
    RETURN(error message);
  IFEND;
END
```

85/12/19

4.0 DESIGN OVERVIEW**4.1.2.2 Define_ipsr_network**

4.1.2.2 Define ipsr network

This routine is called by the Command ME command processor interface task to process the Define_IPSR_network command. This command allows the network operator to add a reachable network to the routing tables. For a definition of the procedure parameters, see the Command ME ERS.

Call format

```
PROCEDURE cmd_define_ipsr_network (  
    parameter_list : ost$string;  
    VAR pvt        : clt$parameter_value_table;  
    VAR c_code     : condition_code;  
    VAR status     : clt$status);
```

Global data accessed

1. The IPSR routine table will be updated.

General Algorithm

```
BEGIN  
    Process the parameters;  
    IF all parameters valid THEN  
        CALL ipsr_add_network;  
        RETURN(appropriate message);  
    ELSE  
        RETURN(error message);  
    IFEND;  
END
```


85/12/19

4.0 DESIGN OVERVIEW

4.1.2.3 Cancel_ipsr_host

4.1.2.3 Cancel ipsr host

This routine is called by the Command ME command processor interface task to process the Cancel_IPSR_host command. This command allows the network operator to delete a host/gateway from the routing tables. For a definition of the procedure parameters, see the Command ME ERS.

Call format

```
PROCEDURE cmd_cancel_ipsr_host (  
  parameter_list : ost$string;  
  VAR pvt       : clt$parameter_value_table;  
  VAR c_code    : condition_code;  
  VAR status    : clt$status);
```

Global data accessed

1. The IPSR routing table will be updated.

General Algorithm

```
BEGIN  
  Process the parameters;  
  IF all parameters valid THEN  
    CALL ipsr_delete_host;  
    RETURN(appropriate message);  
  ELSE  
    RETURN(error message);  
  IFEND;  
END
```

85/12/19

4.0 DESIGN OVERVIEW

4.1.2.4 Cancel_ipsr_network

4.1.2.4 Cancel_ipsr_network

This routine is called by the Command ME command processor interface task to process the Cancel_IPSR_network command. This command allows the network operator to delete a network from the routing tables. For a definition of the procedure parameters, see the Command ME ERS.

Call format

```
PROCEDURE cmd_cancel_ipsr_network (  
  parameter_list : ost$string;  
  VAR pvt       : clt$parameter_value_table;  
  VAR c_code    : condition_code;  
  VAR status    : clt$status);
```

Global data accessed

1. The IPSR routing table will be updated.

General Algorithm

```
BEGIN  
  Process the parameters;  
  IF all parameters valid THEN  
    CALL ipsr_delete_network;  
    RETURN(appropriate message);  
  ELSE  
    RETURN(error message);  
  IFEND;  
END
```

4.0 DESIGN OVERVIEW

4.1.2.5 Process_status_command

4.1.2.5 Process status command

The IPSR module allows the network operator to examine the contents of the various routing tables. The command processor provided is described in this section. For a definition of the parameters, see the Command ME ERS.

Call format

```
PROCEDURE cmd_display_ipsr_table_status (  
  parameter_list : ost$string;  
  VAR pvt       : clt$parameter_value_table;  
  VAR c_code    : condition_code;  
  VAR status    : clt$status);
```

Global data accessed

1. All of the routing tables will be accessed.

General Algorithm

```
BEGIN  
  Process the parameters;  
  IF all parameters valid THEN  
    Get the routing table address;  
    Build the response buffer;  
  IFEND;  
END
```

85/12/19

4.0 DESIGN OVERVIEW4.1.3 INTERNAL_ROUTINES

4.1.3 INTERNAL_ROUTINES

The routines described in this section are internal to the IPSR module and are not provided to the user. This small set of routines will be called by other routines in the IPSR module.

4.1.3.1 Process 3a data

The address of this routine is provided to the 3A module to allow the presentation of data indications. However, the IPSR module should never receive any data from 3A. The routine will therefore discard any data that it should happen to receive and a log message will be generated.

Call format

```
PROCEDURE process_3a_data (
  multicast      : BOOLEAN;
  receive_netid  : net_id_type;
  sending_sysid  : sys_id_type;
  VAR datagram   : buf_ptr);
```

| | | |
|---------------|----|---|
| multicast | In | This flag will be TRUE if the datagram was sent as a broadcast datagram. |
| receive_netid | In | This is the network identifier of the network solution that the datagram was received on. |
| sending_sysid | In | This the system identifier of the system that transmitted the datagram. |
| datagram | In | This is a pointer to the system buffer which contains the datagram. |

CONTROL DATA PRIVATE

4.0 DESIGN OVERVIEW
4.1.3.1 Process_3a_data

Global data accessed

1. None.

General Algorithm

```
BEGIN
  Release the system buffer;
  Log as an internal error;
END
```

85/12/19

4.0 DESIGN OVERVIEW

4.1.3.2 Process_3a_status

4.1.3.2 Process 3a status

The address of this routine will be provided to the 3A module for the presentation of status indications. The status indications will be used by the IPSR module to keep the routing tables up to date as much as possible.

Call format

```
PROCEDURE receive_3a_status (  
    nib_ptr : nib_type);
```

nib_ptr In This is a pointer to the
 information block of the network
 whose status has changed.

Global data accessed

1. The network/host table will be updated.

General Algorithm

```
BEGIN  
    Search for the specified network;  
    IF the network is found THEN  
        Update the status;  
    ELSE  
        CALL ipsr_add_network;  
    IFEND;  
END
```

85/12/19

4.0 DESIGN OVERVIEW
4.2 DATA STRUCTURES

4.2 DATA STRUCTURES

4.2.1 CONSTANTS AND TYPES

The IPSR module uses a number of ordinals. The status returned by its routines is an ordinal, and there are other identifiers that are defined as ordinals. The CYBIL definition of the various ordinals follows.

CONST

```
unreachable_hop = -1,

timeout_threshold = 10,

ipsr_net_unreachable = 20,
ipsr_source_quench = icmp_source_quench,
ipsr_redirect = icmp_redirect,
ipsr_time_exceeded = icmp_time_exceeded;
```

TYPE

```
ipsr_status_type = (
    ipsr_successful,          { Function completed.
    ipsr_host_exists,        { The host is in the list.
    ipsr_insufficient_resources, { Memory alloc. error.
    ipsr_invalid_address,    { Illegal host address.
    ipsr_invalid_owner,     { Owner parameter incorrect.
    ipsr_invalid_type,      { Illegal host type.
    ipsr_network_exists,    { Network already exists.
    ipsr_unable_to_route,   { No route is available.
    ipsr_unknown_gateway,   { Gateway not in the list.
    ipsr_unknown_host,      { No such host in the list.
    ipsr_unknown_network), { No such net in the list.

ipsr_host_type = (
    ip_host,          { IP host on an IP network.
    cdc_host,         { CDC host in a CDC Catenet.
    ip_gw,           { IP gateway on an IP network.
    cdc_gw,          { IP gateway in a CDC catenet.
    local,           { IP address of this gateway.
    none),
```

85/12/19

4.0 DESIGN OVERVIEW

4.2.1 CONSTANTS AND TYPES

```
ipsr_icmp_ind = (  
    ipsr_net_unreachable, ipsr_source_quench,  
    ipsr_redirect,        ipsr_time_exceeded);  
  
ipsr_net_owners = (  
    all,                  { Everyone owns this entry.  
    ipsr,                 { The routing module owns it.  
    egp,                 { The External Gateway owns it.  
    igp),                { The Internal Gateway owns it.  
  
ipsr_connect_status = (  
    unknown,             { The status is unknown.  
    available,          { Status up and available.  
    unavailable,        { Status up but unavailable.  
    down),              { Status down.  
  
ipsr_network_type = (  
    direct,              { Directly connected.  
    remote);            { Connected thru a gateway.
```


85/12/19

 4.0 DESIGN OVERVIEW
 4.2.2 NETWORK/HOST TABLE

4.2.2 NETWORK/HOST TABLE

The information kept by the routing module includes networks and hosts/gateways. The structure used is a nested binary table. The first level table is keyed by network number. Each entry describes a known network. IF the network is directly connected then there will be a field which points to a second level table which contains all of the hosts/gateways that reside on the network; the host number being used as the key.

CYBIL data definitions

```

TYPE
  ipsr_network_info = RECORD
    status : ipsr_connect_status,
    CASE net_type : ipsr_network_type OF
      =direct=
        local_address : ip_address,
        max_data_size : INTEGER,
        3a_network    : net_id_type,
        3a_system     : sys_id_type,
        host_root     : root,
      =remote=
        next_hop      : ip_address,
        hop_count     : INTEGER,
        owner         : ipsr_net_owners,
        timeouts      : INTEGER,
    CASEEND,
  RECCEND,

  ipsr_host_info = RECORD
    host_type : ipsr_host_type,
    egp_active : BOOLEAN,
    igp_active : BOOLEAN,
    3a_system  : sys_id_type,
    status     : ipsr_connect_status,
  RECCEND;

VAR
  default_gateway : ip_address,
  ipsr_routing_table : root;

```

CONTROL DATA PRIVATE

85/12/19

4.0 DESIGN OVERVIEW
4.2.2 NETWORK/HOST TABLE

Data Field Descriptions

default_gateway This is the index of the gateway that datagrams are sent to when they are not being forwarded and the destination network is not contained in the network table.

egp_active This flag is set TRUE if the addressed host supports the DoD External Gateway Protocol.

hop_count This is the number of gateways that a datagram will pass through to reach a remote network.

host_type This is the type of host.

host_root This is a pointer to a directly connected networks host table.

igp_active This flag is set TRUE if the addressed host supports an Internal Gateway Protocol.

ipr_routing_table This is the root of the routing table.

local_address This is the IP address of the host/gateway.

max_data_size This is the maximum size for a datagram that is sent out on a particular network.

next_hop This is the IP address of the gateway that data for a remote network should be sent to.

owner This is the owner of a specific network entry.

status This is the status of a specific network or host.

timeouts This is the number of timeout indications that have been received for datagrams sent to a remote network through a particular gateway.

3a_network This is the 3A network id of the directly connected IP network.

4.0 DESIGN OVERVIEW

4.2.2 NETWORK/HOST TABLE

3a_system

This is the 3A system id of a host on a directly connected network.

Creation/Modification

1. The network/host table will be initialized as empty.
2. The six service routines will update the table.

85/12/19

4.0 DESIGN OVERVIEW
4.3 INITIALIZATION

4.3 INITIALIZATION

The IPSR module does not run as a separate task. The data structures that are used by the IPSR module must be allocated dynamically and SAPs must be opened with the 3A module and the Status module. In order to perform initialization, a flag will be kept to indicate if initialization has been completed. Each routine called from outside the module will check the flag and perform initialization if the flag is not set. So the first call to the IPSR module will force initialization.

85/12/19

4.0 DESIGN OVERVIEW

4.4 DESIGN CRITERIA AND ALTERNATIVES

4.4 DESIGN CRITERIA AND ALTERNATIVES

The design of the IPSR module is very important to all users of the IP module due to the fact that each datagram processed by the IP module will require a call to the IPSR module. The following performance goals have been recognized in the design.

1. Since the routing routine must be called for each datagram, all data tables are designed to optimize the searches needed to determine a route.
2. Because routing table updates are off-line from datagram traffic, the work done by these functions is considered to have low priority.

The general design of the IPSR module is affected by the following functional goals.

1. No set of input values should cause the IPSR module to abort in an uncontrolled manner.
2. Access to the routing tables by outside code should be controlled as much as possible.

85/12/19

| | |
|--|------|
| 1.0 INTRODUCTION | 1-1 |
| 2.0 REFERENCES | 2-1 |
| 3.0 ENVIRONMENT | 3-1 |
| 3.1 HARDWARE | 3-1 |
| 3.2 SOFTWARE | 3-1 |
| 3.2.1 INTERNET PROTOCOL MODULE | 3-1 |
| 3.2.2 3A INTRANET MODULE | 3-1 |
| 3.2.3 STATUS COMMAND PROCESSOR | 3-2 |
| 3.2.4 EXECUTIVE COMMON ROUTINES | 3-2 |
| 4.0 DESIGN OVERVIEW | 4-1 |
| 4.1 FUNCTIONAL STRUCTURE | 4-4 |
| 4.1.1 SERVICE ROUTINES | 4-4 |
| 4.1.1.1 Add_host | 4-4 |
| 4.1.1.2 Add_network | 4-7 |
| 4.1.1.3 Delete_host | 4-9 |
| 4.1.1.4 Delete_network | 4-11 |
| 4.1.1.5 Get_route | 4-13 |
| 4.1.1.6 Process_icmp_data | 4-16 |
| 4.1.1.7 Search_host_table | 4-18 |
| 4.1.1.8 Search_network_table | 4-20 |
| 4.1.2 COMMAND PROCESSORS | 4-22 |
| 4.1.2.1 Define_ipsr_host | 4-22 |
| 4.1.2.2 Define_ipsr_network | 4-24 |
| 4.1.2.3 Cancel_ipsr_host | 4-25 |
| 4.1.2.4 Cancel_ipsr_network | 4-26 |
| 4.1.2.5 Process_status_command | 4-27 |
| 4.1.3 INTERNAL ROUTINES | 4-28 |
| 4.1.3.1 Process_3a_data | 4-28 |
| 4.1.3.2 Process_3a_status | 4-30 |
| 4.2 DATA STRUCTURES | 4-31 |
| 4.2.1 CONSTANTS AND TYPES | 4-31 |
| 4.2.2 NETWORK/HOST TABLE | 4-33 |
| 4.3 INITIALIZATION | 4-36 |
| 4.4 DESIGN CRITERIA AND ALTERNATIVES | 4-37 |