

# Configure DHCP

---

## OPERATING INSTRUCTIONS

## **Copyright**

© Ericsson AB 2017, 2018. All rights reserved. No part of this document may be reproduced in any form without the written permission of the copyright owner.

## **Disclaimer**

The contents of this document are subject to revision without notice due to continued progress in methodology, design and manufacturing. Ericsson shall have no liability for any error or damage of any kind resulting from the use of this document.

## **Trademark List**

All trademarks mentioned herein are the property of their respective owners. These are shown in the document Trademark Information.



# Contents

<b>1</b>	<b>Introduction</b>	<b>1</b>
1.1	Prerequisites	1
1.2	Relation Information	2
<b>2</b>	<b>DHCP Configuration</b>	<b>3</b>
2.1	Creating DHCPv4 Server	3
2.2	Creating DHCP Policy Objects	4
2.3	Creating Option82 Objects	10
2.4	Overlapping Lease Pools	11
2.5	CLF Interface Specifications	13
2.6	DDNS	16
2.7	Failover	19
2.8	Load Balancing	22
2.9	Generating Client Host Names	23
2.10	Authentication of DHCP Clients	25
2.11	User-Defined Options	28
2.12	P-CSCF Discovery	28
2.13	User ID Provisioning	29
2.14	Server Reconfiguration	30
<b>3</b>	<b>DHCP Operations</b>	<b>33</b>
3.1	Starting and Stopping Servers	33
3.2	Updating Servers	33
3.3	Placing Server in Partner-Down Mode	34
3.4	Searching for Lease	35
	<b>Reference List</b>	<b>37</b>





# 1 Introduction

This document describes how to configure IPWorks DHCPv4.

## 1.1 Prerequisites

This section states the prerequisites that must be fulfilled.

- Intermediate Linux and UNIX skills
- Concepts, terminologies, and telecommunication abbreviations, such as TCP/IP, packet data networks, and SC/PL node.
- An Ericsson Command-Line Interface (ECLI) session in Exec mode is in progress.

### 1.1.1 Documents

Before starting this procedure, ensure that the following documents are available:

- For more information about the basics and concepts regarding the configuration management of IPWorks, refer to *IPWorks Configuration Management*, Reference [1].
- For more information about the objects configured through IPWorks CLI (ipwcli), refer to *IPWorks DHCP Parameter Description*, Reference [2].
- This document only introduces the most commonly used configuration scenarios, for complete information about the objects configured through ECLI, refer to *Managed Object Model (MOM)*, Reference [3].

### 1.1.2 Tools

Not applicable.

### 1.1.3 Conditions

Before starting this procedure, the following conditions must apply:

- IPWorks installation is completed.
- Each functionality, associated with a specific license, must be valid and running normally in the license server.



For more information about IPWorks license related information, refer to *License Management*, Reference [4].

- Storage Server is started.
- IPWorks DHCPv4 must be initially configured.

## 1.2 Relation Information

Trademark information, typographic conventions, and definition and explanation of abbreviations and terminology can be found in the following documents:

- *Trademark Information*, Reference [5]
- *Typographic Conventions*, Reference [6]
- *Glossary of Terms and Acronyms*, Reference [7]



## 2 DHCP Configuration

This section provides the following topics to guide the configuration personnel how to configure DHCPv4:

- Creating DHCPv4 Server
- Creating DHCP Policy Objects
- Creating Option82 Objects
- Overlapping Lease Pools
- DDNS
- Failover
- Load Balancing
- Generating Client Host Names
- Authentication of DHCP Clients
- User-Defined Options
- P-CSCF Discovery
- User ID Provisioning
- Server Reconfiguration

**Note:** When the server is configured by the CLI, the changes are applied to the network after updating the server and restarting the service.

### 2.1 Creating DHCPv4 Server

The commands below show an example of the creation of a DHCPv4 server object *dhcpv4server*. No area is specified, so it uses the default area for locating policies. Since no primary is specified, this server is assumed to be a standalone server (for example, not part of a failover pair).

```
IPWorks> create dhcpv4server dhcp0 -set dnsname=dhcp0.example.com -set address=10.0.0.1
```

```
1 object(s) created.
IPWorks> list
[DhcpV4Server dhcp0]
  Partition: active
  Name: dhcp0
  Address: 10.0.0.1
```



```
PrimaryAddress: 10.0.0.1
DnsName: dhcp0.example.com
PrimaryDnsName: dhcp0.example.com
Type: DhcpV4
Area: default
Filename: dhcpd.conf
ExportNeeded: true
EnableAuthentication: false
```

**Note:** If IPWorks DHCPv4 server is installed on a system in the network, it does auto-register itself by default when the Server Manager for that server first starts up. It creates a DHCPv4 server object in the database to correspond to the active server and is defined with the IP addresses and domain names for that system. Use of this feature allows the user to have these objects automatically, but automatically created objects must be reviewed to ensure that they reflect the configuration desired for the server. The Server Manager uses the hostname on which the server is running as the server name.

## 2.2 Creating DHCP Policy Objects

This section describes how to create DHCP policy objects.

### 2.2.1 Creating Subnet

The following example creates 4 *subnets*.

The first subnet represents a class B subnet that contains the other three subnets.

For example:

```
IPWorks> create subnet 10.1.0.0/16
```

```
1 object(s) created.
```

The second subnet is a class C subnet. When using the standard Internet naming syntax to name the subnet, all the fields of the subnet can be automatically computed and output after the subnet is created.

For example:

```
IPWorks> create subnet 10.1.1.0/24
```

```
1 object(s) created.
```

```
IPWorks> list
```

```
[Subnet 10.1.1.0/24]
Partition: active
Area: default
Name: 10.1.1.0/24
```



```
Address: 10.1.1.0
Mask: 255.255.255.0
MaskLength: 24
FirstSortKey: 167837952
LastSortKey: 167838207
BroadcastAddress: 10.1.1.255
IsDistinct: true
Server: none
```

The last two subnets are created using an alternate naming scheme. Both subnets use the CIDR based addressing mode, which split a class C subnet into two smaller subnets, and both subnets contain 126 usable addresses.

For example:

```
IPWorks> create subnet subnet2a -set address=10.
1.2.0 -set masklength=25
1 object(s) created.
IPWorks> create subnet subnet2b -set address=10.1.2.128 -set maskle
1 object(s) created.
IPWorks> modify subnet 10.1.1.0/24 -set server=dhcp0
Working on 1 object(s).
1 object(s) were updated.
```

Of the four subnets, only the second one is activated to be used in the DHCP server (by setting the Server field to the name of a DHCP server).

## 2.2.2 Creating Pool

This example shows the creation of a *pool*. Pools must be contained in a subnet that already exists (the pool will be updated with a field that identifies that subnet).

```
IPWorks> create pool pool1 -set addressrange=10.1.1.1-100
1 object(s) created.
IPWorks> modify -set server=dhcp0
Working on 1 object(s).
1 object(s) were updated.
IPWorks> list
[Pool pool1]
Partition: active
Name: pool1
Subnet: 10.1.1.0/24
AddressRange: 10.1.1.1-100
Server: dhcp0
Area: default
V4option: pool-index 648612050
```



**Note:** Though a pool can be created without associating it with a server, it cannot be used until it is associated with a DHCP Server.

The pool is not activated by default when it is created.

This example shows the creation of a second pool, but this pool is associated with a DHCP server when it is created. It is active on the network as soon as that DHCP server is updated. Associating the pool with a server, automatically associates the subnet that contains the pool with that server.

```
IPWorks> create pool pool2 -set addressrange=10.1.2.1-10 -set server=dhcp0
1 object(s) created.
IPWorks> list subnet subnet2a
[Subnet subnet2a]
Partition: active
Area: default
Name: subnet2a
Address: 10.1.2.0
Mask: 255.255.255.128
MaskLength: 25
FirstSortKey: 167838208
LastSortKey: 167838335
BroadcastAddress: 10.1.2.127
IsDistinct: true
Server: dhcp0
```

The pools have to be updated so that pool1 is reserved for known clients and pool2 is reserved for unknown clients.

```
IPWorks> modify pool pool1 -set allowedclient=known
Working on 1 object(s).
1 object(s) were updated.
IPWorks> modify pool pool2 -set allowedclient=unknown
Working on 1 object(s).
1 object(s) were updated.
IPWorks> list
[Pool pool2]
Partition: active
Name: pool2
Subnet: subnet2a
AddressRange: 10.1.2.1-10
AllowedClient: unknown
Server: dhcp0
```

### 2.2.3 Creating Link

In the following example, a link is created and then two of the previously created subnets (subnet2a and 10.1.1.0/24) are associated with that



link. The configuration of the link is then examined, as well as the associated subnets and pools.

```
IPWorks> create link floor1_lan
1 object(s) created.
IPWorks> modify subnet subnet2a -set link=floor1_lan
Working on 1 object(s).
1 object(s) were updated.
IPWorks> modify subnet 10.1.1.0/24 -set link=floor1_lan
Working on 1 object(s).
1 object(s) were updated.
IPWorks> list link -format=conf
```

```
shared-network "floor1_lan" {

    # Subnet(s):
    # Subnet 10.1.1.0/24
    subnet 10.1.1.0 netmask 255.255.255.0 {
    # Pool(s):
    # Pool pool1
    pool {
        pool-name "pool1";
        pool-index 648612050;
        range 10.1.1.1 10.1.1.100;
        allow known clients;
    }
    }
    # Subnet subnet2a
    subnet 10.1.2.0 netmask 255.255.255.128 {
    # Pool(s):
    # Pool pool2
    pool {
        pool-name "pool2";
        pool-index 648843534;
        range 10.1.2.1 10.1.2.10;
        allow unknown clients;
    }
    }
}
```

## 2.2.4 Creating Client

This example creates a *client* policy for a printer that will always be issued the same address. Assigning fixed addresses with DHCP ensures that certain devices always have the same address, and makes it easy to reconfigure the addresses instead of reconfiguring the devices:

```
IPWorks> create client printer1 -set macaddress="01:0
3:04:05:06:07" -set fixedaddress=10.3.1.11
```



```
1 object(s) created.
```

This example shows how to declare a client so that it will be considered as a known client. This example uses the DHCP client identifier instead of the MAC address to identify the client when the client requests an address from the server:

```
IPWorks> create client bobs_laptop -set client
identifier="bobs_laptop"
1 object(s) created.
```

This example shows a declaration for a client that is not allowed to lease an address. This feature is only useful when the faulty system is identified by its MAC address:

```
IPWorks> create client bogus-host -set macaddress="01
:01:01:01:01:01" -set denybooting=true
1 object(s) created.
```

To see what the client declarations look like in the DHCP servers configuration file:

```
IPWorks> list clients -format=conf
host bobs_laptop {
    option dhcp-client-identifier "bobs_laptop";
}
host bogus-host {
    hardware ethernet 01:01:01:01:01:01;
    deny booting;
}
host printer1 {
    hardware ethernet 01:03:04:05:06:07;
    fixed-address 10.3.1.11;
}
```

## 2.2.5 Creating ClientClass

This example uses the `if` syntax to determine whether clients are the members of a client class. This client class limits the number of RAS clients. So five clients at most can be able to lease addresses at a time:

```
IPWorks> create clientclass rasclients -set leaselimit=5
-set match="if substring(option dhcp-client-iden
tifier, 0, 3) = \"RAS\""
1 object(s) created.
IPWorks> list -format=conf
class "rasclients" {
```



```
match if substring (option dhcp-client-identifier, 0, 3)
= "RAS";
lease limit 5;
}
```

This example specifies a specific option to be matched. In this case it is a special option that gets set by a DHCP relay. This class definition is intended to be used in an environment where DHCP is being used to issue addresses to be used with a cable modem (which acts as a DHCP relay). This client class automatically spawns subclasses (one for each cable modem) — each of which will have a lease limit of three addresses. This limits each customer to three addresses per cable modem:

```
IPWorks> create clientclass cablemodem -set match="if
substring(option dhcp-client-identifier,0,3) = \"RAS\"
-set spawn="option agent.remote-id" -set leaselimit=3
```

1 object(s) created.

```
IPWorks> list -format=conf
```

```
class "cablemodem" {
match if substring (option dhcp-client-identifier, 0, 3)
= "RAS";
spawn with option agent.remote-id;
lease limit 3;
}
```

## 2.2.6 Creating Subclass

In this example, a client class object *clientclass* that matches the DHCP user class option is created, as well as predefined subclasses that associate the correct printer with the client.

```
IPWorks> create clientclass userclass -set
match="userclass"
1 object(s) created.
IPWorks>create subclass userclass "engineering" -set option="lpr-se
1 object(s) created.
IPWorks>create subclass userclass "sales" -set option="lpr-server s
1 object(s) created.
IPWorks> list clientclass userclass -format=conf
class "userclass" {
  match option user-class;
}
subclass "userclass" engineering {
  option lpr-servers engprinter1;
}
subclass "userclass" sales {
  option lpr-servers salesprinter1;
```



```
}
```

## 2.3 Creating Option82 Objects

The two Option82 objects *dhcpv4option82format* and *dhcpv4option82iprange* are created together as a pair due to the interrelationship.

For example:

```
IPWorks> create dhcpv4option82format juniper_erx
-set category=0;suboptionid=1;informat="^(.*) :ATM
([0-9]{1,2})/([0-9]{1,2}).?[0-9]*:([0-9]{1,3}).([0-9]
){1,5})$";outformat="$1#$2#$3#$4#$5"
IPWorks> create dhcpv4option82iprange iprange1 -set
address=10.170.1.0 -set masklength=25;server=dhcp
0;option82format=juniper_erx
```

If the **name** field of the object *dhcpv4option82iprange* adopts the standard Internet naming syntax, other fields can be computed automatically, which is similar to creating a subnet.

For example:

```
IPWorks> create dhcpv4option82iprange 10.170.1.0/25 -set
server=dhcp0;option82format=juniper_erx
1 object(s) created.
IPWorks> list
[Dhcpv4Option82IPRange 10.170.1.0/25]
  Partition: active
  Name: 10.170.1.0/25
  Address: 10.170.1.0
  Mask: 255.255.255.128
  MaskLength: 25
  Server: dhcp0
  Option82Format: juniper_erx
```



**Note:** The **name** allsubnet is used to define a special IP range that includes all the subnets. For example:

```
IPWorks> create dhcpv4option82iprange allsubnet -set
server=dhcp0;option82format=juniper_erx
1 object(s) created.
IPWorks> list
[Dhcpv4Option82IPRange allsubnet]
Partition: active
Name: allsubnet
Address: 0.0.0.0
Mask: 0.0.0.0
MaskLength: 0
Server: dhcp0
Option82Format: juniper_erx
```

## 2.4 Overlapping Lease Pools

In IPWorks, it is possible to specify the lease pools with partial or total overlapping of the IP addresses. Overlapping is applied when many sets of client machines are to be leased with limited set of address pools, thereby maintaining the authenticity.

The following condition should be full filled:

- The overlapping IP addresses that clients request must belong to different physical networks, which are separated by relay agents, routers, or other network devices.
- All the overlapping pools must also belong to the same logical subnet.

With overlapping, the lease pools can have identical IP addresses. To avoid address space conflicts, the lease pools are configured to allow or deny different client classes. Figure 1 shows how multiple requests with identical IP addresses are received and addressed appropriately, by defining the vendor classes distinctly.

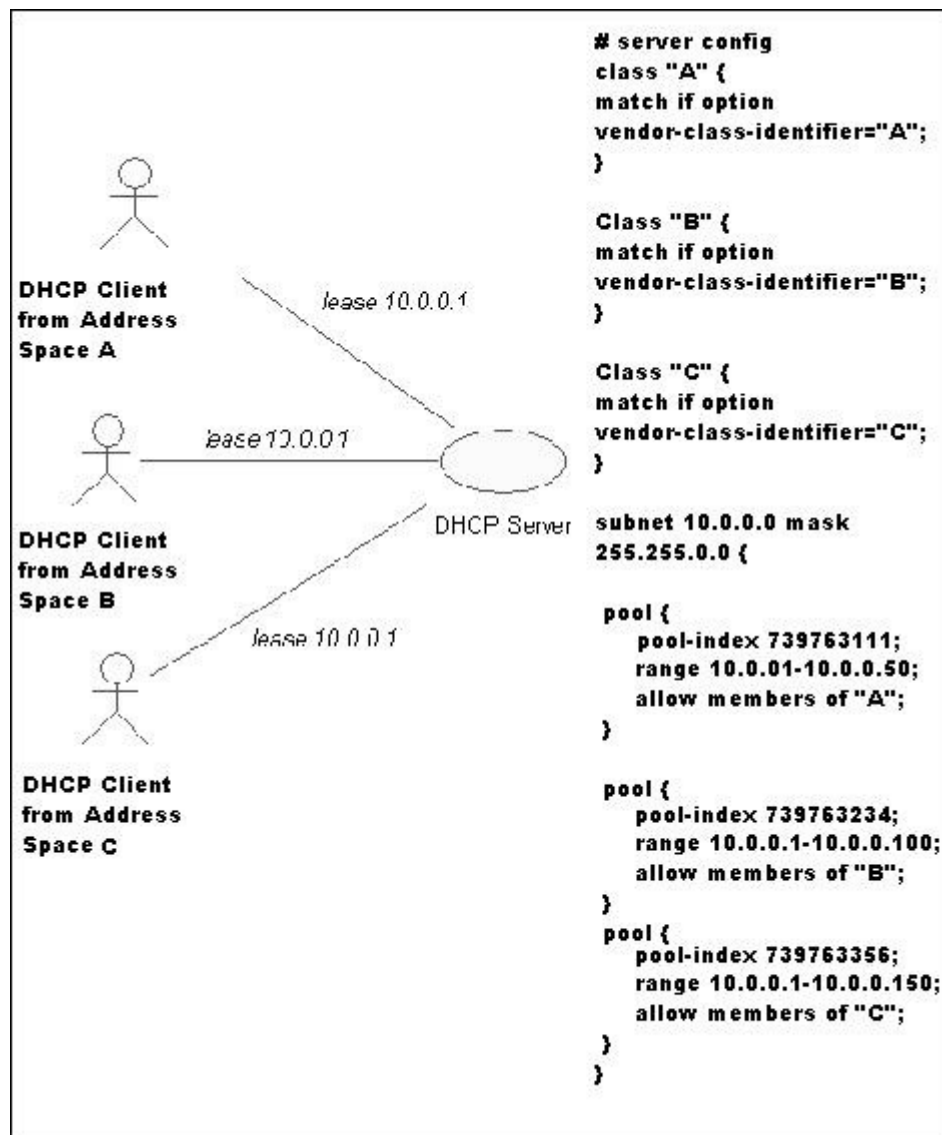


Figure 1 Overlapping Lease Pools

The following values must be distinct within a `vendor-class`:

- the value of *MACAddress*
- the value of *ClientIdentifier*
- the value of *UserClass*

Otherwise, for a given pool, the `vendor-class-identifier` must be unique.

For example, as shown in Figure 1, pools 1, 2, and 3 have overlapped address range 10.0.0.1-10.0.0.50. Client class A can be defined using the `vendor-class-identifier` as follows:



```
IPWorks> create clientclass Aclients -set lease1
imit=5 -set match="if substring(option vendor-cl
ass-identifier,0,1)= \"A\""
```

1 object(s) created.

Pool 1 is specified to allow client class A as follows:

```
# Pool pool1
pool {
    pool-index 7397637111;
    range 10.0.0.1 10.0.0.50;
    allow Aclients;
}
```

Similarly, class B and C are defined. Pool 2 and Pool 3 are specified to allow class B and C respectively.

**Note:** By default, if a class is not allowed, it is considered as denied for a given pool. However, it is recommended to define a deny rule for class A on the overlapping pools, namely Pool 2 and Pool 3.

## 2.5 CLF Interface Specifications

### 2.5.1 Messages

The following four routers are supported by NACF by default. This is due to how they form Option82 and the way Option82 is decoded and reformatted by NACF to form CLID and Remoteld:

- Alcatel
- Cisco
- Juniper
- Red Back

Two ways are available to define the IP range, the format, and the relationship between them.

#### Use default formats

1. Execute the CLI command file.

```
IPWorks> create dhcpv4option82format juniper_erx
-set category=0;suboptionid=1;informat="^(.+):ATM
([0-9]{1,2})/([0-9]{1,2}).?[0-9]*:([0-9]{1,3}).([0-
9]{1,5})$";outformat="$1#$2#$3#$4#$5"
```

1 object(s) created.

```
IPWorks> create dhcpv4option82format redback -set category=0;sub
```



```
1 object(s) created.
IPWorks> create dhcpv4option82format alcatel -set category=0;suboptio
1 object(s) created.
IPWorks> create dhcpv4option82format cisco -set category=1;suboptio
1 object(s) created.
IPWorks> create dhcpv4option82format giga -set category=0;suboptio
1 object(s) created.
IPWorks>
```

2. Use the list command to find the **name** values of the default formats.

```
IPWorks> list dhcpv4option82format
[Dhcpv4Option82Format alcatel]
Partition: active
Name: alcatel
Category: 0
Suboptionid: 1
Informat: ^(.*) atm 1/1/0?([1-9]{1,2})/0?([1-9]{1,2}):([0-9]).([0-9]
Outformat: $1#$2#$3#$4#$5

[Dhcpv4Option82Format cisco]
Partition: active
Name: cisco
Category: 1
Suboptionid: 1
Informat: $1(5):$2(6):$3(7):$4(8):$5(9.1,9.4):$6(9.6,9.8):$7(10):$8
Outformat: $1.$2.$3.$4#$5#$6#$7#$8

[Dhcpv4Option82Format juniper_ern]
Partition: active
Name: juniper_ern
Category: 0
Suboptionid: 1
Informat: ^(.+):ATM ([0-9]{1,2})/([0-9]{1,2}).?[0-9]*:([0-9]{1,3})
Outformat: $1#$2#$3#$4#$5

[Dhcpv4Option82Format redback]
Partition: active
Name: redback
Category: 0
Suboptionid: 1
Informat: ^(.+):([0-9]{1,2})/([0-9]{1,2}) vpi-vci ([0-9]{1,3}) ([0-9]
Outformat: $1#$2#$3#$4#$5

[Dhcpv4Option82Format remoteid]
Partition: active
Name: remoteid
Category: 0
Suboptionid: 2
Informat: ^(.+)!(!.+) $
```



Outformat: \$1!\$2

3. Create `dhcpv4option82iprange` to define the IP range and the relationship between the IP range and the format.

### Use customized formats.

See Section 2.3 Creating Option82 Objects on page 10.

After defining a group of `dhcpv4option82format` objects, do not apply the format rules of different types (CLID or RemoteId, Binary or ASCII) to a specific IP range. For example, in the default format rules, `remoteid` must not be mixed with `alcatel`, `cisco`, `juniper_erx`, and `redback` because `remoteid` is for RemoteId and the others are for CLID; `cisco` must not be mixed with `alcatel`, `juniper_erx` and `redback` because `cisco` is for binary category and the others are for ASCII category.

## 2.5.2 Alternate and Exceptional Scenarios

If the CLF Interface Enable parameter is set to zero (0), NACF will not establish connection towards CLF when NACF is started. However, if the parameter is set to one (1), since it's a real time parameter, NACF will immediately active the interface and will try to establish connection towards CLF.

## 2.5.3 Configuring NACF-CLF Interface Parameter

1. Enable CLF Interface

Enable *NACFService* and configure the communication mode:

```
>ManagedElement=<Node Name>,IpworksFunction=1,IPWork
sDHCPRoot=1,NACFService=1
```

```
(NACFService=1)>configure
```

```
(config-NACFService=1)> clfInterfaceEnable=true
```

```
(config-NACFService=1)> clfOperateMode=0
```

```
(config-NACFService=1)> commit
```

2. Configure CLF Address

To activate the NACF-to-CLF interface, the parameter `CLF Address` needs to be configured in the NACF configuration file (`dhcpd.conf`). The `CLF address` is an IPv4 address.

The multiple pools are able to serve the same addressing zone, but they must be configured with mutually exclusive address ranges.



The CLI commands for the CLF `Address` parameter are as follows:

```
IPWorks> select dhcpv4server <server name>

1 object selected

IPWorks> modify -add option="clf-address <CLF IP
Address>"

1 object updated
```

### 3. Configure Multiple IAZ

Multiple IP Address Zone (IAZ) is used to uniquely identify an IP Address in the CLF, and it is applied when the overlapping IP address spaces are used over the different access networks.

The IAZ is configured at the pool level. In push mode, both synchronous and asynchronous modes, NACF forwards the IP address and its related IAZ to CLF. In pull mode, NACF keeps line information together with the IAZ and IP address. If IAZ and IP address are correct, the line information is sent back to CLF.

**Note:** The overlapped pools belong to the same DHCPv4 server must have different IAZ names.

The CLI commands for the `clf-address-zone` are as follows:

```
IPWorks> select pool <pool name>

1 object selected

IPWorks> modify -add option="clf-address-zone <IP Address
zone>

1 object updated
```

## 2.6 DDNS

With DHCP, Internet addresses are dynamically assigned to clients. Dynamic DNS (DDNS) is used to register these addresses and the name of the associated client, dynamically to DNS. IPWorks DNS server and DHCP server support DDNS. The most effective way to use DDNS updates is to create a new zone and use only dynamic updates to create the resource records in that zone. The main advantages of using a separate zone for dynamic updates are:

- Names do not conflict with static or non-dynamic names. This prevents possible naming conflicts and problems, such as registering as the organization's web server.
- Zone updates are easier to use. If a zone contains both static and dynamic names, zone updates are complex because it is not always clear which



operations (static modifications or dynamic updates) changed the zone and when. For example, if a zone file is being manually edited or exported while dynamic updates are occurring, the dynamic updates might be lost.

**Note:** DDNS updates are not done for clients with fixed addresses. The DHCP server assumes that these are fixed addresses and the DNS entries are static.

If the user wants to enable DDNS:

- Create a dynamic zone in DNS Server for a PTR Record update, such as `170.10.in-addr.arpa`.
- In IPWorks, edit the `/cluster/etc/cluster.conf` file, add the DNS for the node PL-3/PL-4 which DHCP server will be deployed, replace the IP for nameserver with the IP for the DNS server, on which DDNS determines to update. For example `/cluster/etc/cluster.conf`

```
node 3 payload PL-3
dns 3 10.175.171.100
```

```
node 4 payload PL-4
dns 4 10.175.171.100
```

- Update the `/etc/resolve.conf` by reloading cluster configuration on PL-3/PL-4, since DHCP server will retrieve the DNS server from the `resolve.conf` file, and update *ARecord* or *PTRRecord* to that server.

```
On SC
SC-1:~ #lde-config --reload --node 3
SC-1:~ #lde-config --reload --node 4

/etc/resolve.conf
PL-3:~ # cat /etc/resolv.conf
#
# /etc//resolv.conf: resolver configuration
#

nameserver 10.175.171.100

# End of file
```

- The following options must be configured:
  - **Domain-name:** Used to construct the domain part of the FQDN for DDNS updates.
  - **Host-name:** Used to construct the local part of the FQDN for DDNS updates.



- **Dynamic-update:** Used to instruct the DHCP server to perform DDNS updates for the client. The DHCP server always updates the PTR resources record for the client.
- **Update-a-record:** Used to instruct the DHCP server to also update the A resource record for the client.

In the following example, DDNS is turned on for one of the lease pools. The host names are generated based on the address assigned to the client (See Section 2.9 Generating Client Host Names on page 23).

```
IPWorks> modify -add option="domain-name dynamic.example
.com" -add option="host-name pc%H" -add option="dynamic
-update true" -add option="update-a-record true"
Working on 1 object(s).
1 object(s) were updated.
IPWorks> list -format=conf

# Pool pool1
pool {
    range 10.1.1.1 10.1.1.100;
    allow known clients;
    default-lease-time 86400;
    option domain-name "dynamic.example.com";
    option host-name "pc%H";
    dynamic-update true;
    update-a-record true;
}Using
```

## 2.6.1 Securing DDNS with TSIGKeys

The IPWorks DHCP server can be configured to use TSIG transaction security in conjunction with dynamic updates to a TSIG-enabled DNS server such as an IPWorks DNS Server. To enable this feature, the `secure-ddns` option is configured (set to true). Then `TSIGKey` used for the DNS Server should be created by creating a `DnsContact` object. A `DnsContact` is an object that defines information to be used in contacting a DNS server. The server that is being contacted does not need to be an IPWorks managed server. To create a `DnsContact` the following information must be provided:

- A **name**: This is any name that is used to remember the `DnsContact`.
- The **address** of the server.
- The **TCP/IP** port on which the server listens.
- The name of a **TSIGKey** used to secure packets sent to the server. If a value is specified for this field, create the key before the contact object is created.



- The DHCP **declarer**: The name of the DHCP servers that use this contact.

**Note:** `DnsContacts` is also used in the configuration of DNS servers, which need to contact other servers, and ActiveSelect Monitors. It is also possible to share a `DnsContact` object among multiple server types, hence there must be caution when working with these objects.

This example illustrates how to secure DDNS communications between the sample DHCP server and a DNS Server. In this example all the lines of the configuration file have been deleted except for the declaration of the key. The name of the key is the address of the DNS server.

```
IPWorks> create tsigkey ns0-key
1 object(s) created.
IPWorks> create dnscontact -set name=ns0 -set address=10.1.0.1 -set
1 object(s) created.
IPWorks> list dhcpserver dhcp0 -format=conf

# Configuration file for DHCP Server dhcp0

...<irrelevant lines removed>...

key 10.1.0.1 {
    key-id "ns0-key";
    algorithm "HMAC-MD5";
    secret "KvW0nt5BW3CKQ7lZ+jS+NQ==";
}
```

## 2.7 Failover

This section discusses the configuration and use of the DHCP failover protocol with IPWorks.

### 2.7.1 Failover in DHCP

There are requirements for using the failover protocol support in IPWorks:

- Two DHCP servers are needed on which the failover partner DHCP server can be deployed. The time difference between the server pair must not be greater than 60 seconds, otherwise both the servers will run in the **Communication Interrupted** mode, instead of **Normal** mode.
- BOOTP or DHCP relay agents are used on the network to forward client broadcasts to both failover partner servers. For most routers, do this by specifying both servers IP addresses in the UDP helper (or BOOTP or DHCP relay) address list. Refer to the documentation for the router vendor for details.



- If a fire wall is in the network between the two IPWorks DHCP servers, TCP ports 647 needs to be opened for communication (these are the default port values and can be altered if needed through the DHCP options for the configuration). Refer to the documentation for the fire wall vendor for details.

## 2.7.2 Failover Load Balancing

The IPWorks DHCP server uses the load balancing option when operating in a failover mode. Load balancing means that each server responds to a subset of the clients. Failover load balancing is only in operation when the servers are in communication with each other; thus if one server fails, the other cannot only handle a subset of clients but all the clients. This support is based on the IETF Load Balancing specification (RFC 3074).

When using failover, the `failover-split` option for the DHCP configuration controls the ratio of clients serviced by the primary server (the secondary server services the remaining clients). The server uses a hash function to map all clients according to their identifiers (or `hlen` field of the DHCP packet is used as the length of the data to be hashed if no Client Identifier option is present) evenly to an HBA (hash bucket assignment) value from 0 to 255. This option should be set only for primary server and the secondary server fetches this option from primary server during startup. The default `failover-split` is 128 which means that clients whose hash values fall into 0-128 are served by the primary server and is the recommended setting. IPWorks also supports load balancing when multiple failover pairs are used. For more details on Load Balancing, see Section 2.8 Load Balancing on page 22.

For example, when the respond time of the primary server towards the requests exceeds the `failover-load-balance-max-seconds`, the secondary server will process those requests, and vice versa. The maximum respond seconds can be specified through `failover-load-balance-max-seconds` option. The default value is 30, which means the secondary server will respond the requests that primary server failed to respond within 30 seconds. This support is based on the IETF Load Balancing specification (chapter 5.3 Delayed Service Parameter, RFC 3074) and DHCP Failover Protocol (chapter 5.5 Operating in NORMAL state, draft-ietf-dhc-failover-12).

In the following example, a primary DHCP server is configured with `failover-split` value 64, it will serve one quarter of the whole clients, and the secondary server will serve the others.

**Note:** Before implementing the following command, the primary and secondary DHCP servers must be specified.

```
IPWorks> modify dhcpv4server dhcp0 -add option
="failover-split 64"
Working on 1 object(s).
1 object(s) were updated.
IPWorks> list dhcpv4server -format=conf
# Configuration file for DHCP Server dhcp0
```



```
...
failover peer "peer" {
primary;
address 10.170.4.45;
port 647;
peer address 10.170.4.43;
peer port 647;
split 64;
}
```

When the load balance is configured, the server will process all the requests received from its designated clients using the load balance algorithm, however, the peer will process the client's request if the respond time of the server exceeds the `failover-load-balance-max-seconds`. The maximum number of seconds can be specified through `failover-load-balance-max-seconds` option.

### 2.7.3 Configuring a Pair of Failover Servers

To configure a pair of failover servers, the servers should be installed using the normal installation procedure for a single DHCP server. Always configure the server as primary.

To enable the failover protocol, the secondary server needs to be modified. Each DHCPv4 Server has a field called `Primary`. This field should only be set on the secondary server in a failover pair. It is set to the name of the primary server. Once it is set, the secondary server will automatically use the configuration information that is associated with the primary server. The primary server will also be automatically updated to include the failover configuration for the secondary server.

Normally the failover protocol uses the primary address associated with a server. This can be customized by setting the `FailoverAddress` field on the server. If this is set then the failover protocol will use this address instead of the primary address for the server.

After the server is updated (for information on Updating Servers, see Section 3.2 Updating Servers on page 33) the failover protocol will be enabled between the servers. No other changes are necessary.

**Note:** When two servers are configured as a failover pair, any time that one server is updated its peer will automatically be updated as well.

In this example a second DHCP server that can be used as the backup server in a failover pair is created. The configuration for the existing server to make sure it contains the appropriate changes to turn on the failover protocol is examined later.

```
IPWorks> create dhcpv4server dhcpbackup -set address=10.0.
0.2 -set dnsname=dhcpbackup.example.com -set primary=dhcp0
```



```
1 object(s) created.
IPWorks> list dhcpserver dhcp0 -format=conf

# Configuration file for DHCP Server dhcp0
...<irrelevant lines removed>...
failover peer "peer" {
    primary;
    address 10.0.0.1;
    port 647;
    peer address 10.0.0.2;
    peer port 647;
}
```

### 2.7.4 Failover Options

There are some failover options (parameters) which can be set for DHCPv4 server when failover pair is deployed.

The following command can be used to find help for all the failover related options:

```
IPWorks> show dhcpv4 options failover*
```

The following command can be used to set the options for DHCPv4 server which is already created.

```
IPWorks> modify DHCPv4server <server name> -add
option=<optionname value>
```

For example:

```
IPWorks> modify DHCPv4server dh1 -add option="failover-s
plit 200"
```

For the options that can be set to configure failover parameters, refer to *IPWorks DHCP Parameter Description*.

## 2.8 Load Balancing

The IPWorks DHCP server can be configured to use load balancing if multiple servers or multiple failover pairs are running. Load balancing means that each server (or failover pair) responds to a subset of the clients instead of one server (or failover pair) responding to all of the clients. A failover pair will further split the subset of assigned clients between the two failover servers (this is failover load balancing and separate from server load balancing).

In order to configure load balancing, the decision to split the clients among the servers must be taken. This is done by using the hba option for each server. This option allows the user to split the overall client space in a variety of ways,



ensure that the clients are split so that all the clients are handled by exactly one DHCP server or one failover pair.

In the following example, the sample is modified by configuring the existing pair of failover servers (dhcp0 and dhcpbackup) to handle half of the client requests and then create a third DHCP server which will handle the other half of the DHCP requests.

**Note:** The steps to configure the rest of the DHCP policies for the new DHCP server have been omitted

```
IPWorks> show dhcpv4 option hba
hba Description: This specifies the load balancing hash bucket entry
...
Server Config:  if <value> = :
                  hba <value>;
IPWorks> select dhcpserver dhcp0
Selected 1 object(s).
IPWorks> modify -add option="hba 1/2"
Working on 1 object(s).
1 object(s) were updated.
IPWorks> create dhcpv4server dhcp3 -set address=10.0.0.3 -set dnsna
1 object(s) created.
```

## 2.9 Generating Client Host Names

The IPWorks DHCPv4 Server supports the automatic generation of host names. To use this feature, host-name options are specified, but the value specified can contain substitution keys that are replaced by context-specific values. Substitution keys are indicated by using the % character in the string value. Using this notation, one (or more) values are substituted. Page 23 describes these values.

**Note:** If the generated host name exceeds 63 characters, it is truncated.

Page 23 shows the Host name Options:

*Table 1 Host Name Options*

Option	Description
%A	First byte of the host's IP address (with 192.168.11.23 returns 192)
%B	Second byte of the host's IP address (with 192.168.11.23 returns 168)
%C	Third byte of the host's IP address (with 192.168.11.23 returns 11)
%D	Fourth byte of the host's IP address (with 192.168.11.23 returns 23)



Option	Description
%H	Host part of the host's IP address; must include the subnet mask option to indicate which part of the IP address represents its host address
%I	Client ID of the host; must be defined for the host %I Client ID of the host, as above, except hyphens (-) are used instead of the usual byte separators in the Client ID value.
%M	MAC address of the host; must be defined for the host
%-M	MAC address of the host, as with %M, except hyphens (-) are used instead of the usual byte separators in the MAC address value.
%N	Host name returned from the client if the host name option is supplied by the client.
%P	Hostname returned from the printable characters which is from client id if the client identifier option is supplied by the client.
%S	Subnet part of the host's IP address; must include the subnet mask option to indicate which part of the IP address represents its subnet.
%-S	Subnet part of the host's IP address, as with %S, except hyphens (-) are used instead of the usual byte separators in the subnet part of the address.

For example, if the value were %N-%M, this identifies the host by the name supplied by the client and adds the host's MAC address, separated by a hyphen. The key values included in the host name option of a lease pool might generate the following host names, each of which is unique:

```
boston-006008A08CAA  
rita-006008A10CAA  
host1-006008A12CAA  
host2-006008A14CAA  
boston-006008A16CAA
```

And, if the value were Host%H-%S for a lease pool with an address range of 192.168.11.6 through 192.168.11.10 and a subnet mask of 255.255.255.0 would generate the following host names:

```
Host6-192-168-11  
Host7-192-168-11  
Host8-192-168-11  
Host9-192-168-11  
Host10-192-168-11
```



## 2.10 Authentication of DHCP Clients

The following features are applied to DHCPv4 servers in order to authenticate DHCP client. The configuration of keys is different from that of DHCP client.

In order to enable the feature, the following steps are required:

1. Authentication level should be configured correctly.
2. The administrator needs to configure each client with a unique key on a DHCP server. These keys are the shared secrets between DHCP clients and servers. These keys are stored in the `dhcpkey.conf` file that is uploaded to the DHCP configuration directory.
3. The `EnableAuthentication` option of `dhcpv4server` should be set to `true`.

**Note:** The `dhcpkey.conf` file is uploaded when DHCP configuration is updated from the CLI: the `EnableAuthentication` option should be set to `true` for this to happen. For more information on the `dhcpkey.conf` file, see Section 2.10.2 Configuring Authentication Keys on page 26.

The operator must ensure that when the authentication level is `MANDATORY` or `OPTIONAL`, `dhcpkey.conf` file is in the configuration directory. Otherwise, the authentication fails.

### 2.10.1 Configuring Authentication Level

Configuring authentication level on the server proceeds as follows:

Login to ECLI.

```
>ManagedElement=<Node Name>,IpworksFunction=1,IPWorksDHCPRoot=1,DHCPv4Service=1
```

```
(DHCPv4Service=1)>configure
```

```
(config-DHCPv4Service=1)>authenticationLevel=MANDATORY
```

```
(config-DHCPv4Service=1)>commit
```

```
(DHCPv4Service=1)>show -v
```

```
DHCPv4Service=1
  arguments="ipw_sig_sp"
  authenticationLevel=MANDATORY
  dhcpv4ServiceId="1"
  EnableAutoReconfig=false <default>
  lowTPSThreshold=0
  reconfigThreshold=0 <default>
```



```
(DHCPv4Service=1)>
```

## 2.10.2 Configuring Authentication Keys

Configuring authentication keys on the server proceeds as follows:

Login to IPWCLI.

```
IPWorks> create dhcpv4authkey -prompt
AuthKeyID? [next] > 1
```

Each authentication key has an ID, which can be a number value.

```
AuthKey? [next] > 0x0b0b0b0b0b0b0b0b0b0b0b0b0b0b0b0b
```

The authentication key can be a hexadecimal number.

```
ClientID? [next] > ISCDhcpClient
```

Identifier of the client whose key is being configured on the server.

```
Server? [next] > tn0-s1
Server: tn0-s1
Server? [next] >
```

Name on the server on which the client keys are being configured.

```
1 object(s) created.
```

The syntax of configuring keys (without prompt) is as follows:

```
IPWorks> create dhcpv4authkey -set AuthKeyID=<keyID>;
AuthKey=<key>; ClientID=<ip_address or MAC_address>;
Server=<server_host_name>
```

## 2.10.3 Configuring EnableAuthentication Option

Before updating the server, verify that *EnableAuthentication* is set to true: this uploads the `dhcpkeys.conf` file to the Storage Server. Without enabling this option, if configuration is updated, the `dhcpkeys.conf` file is NOT written to the Storage Server.

```
IPWorks> list dhcpv4server
[DhcpV4Server tn0-s1]
Partition: active
```



```
Name: tn0-s1
Type: DhcpV4
Area: default
Filename: dhcpd.conf
Export Needed: True
EnableAuthentication: false
```

If `EnableAuthentication` is *false*, then set it to true as follows:

```
IPWorks> modify dhcpv4server tn0-s1 -set enableAuthentication=true
Working on 1 object(s).
1 object(s) were updated.
```

Now, issue the update command to configure the keys on the server:

```
IPWorks> update
Exported Authentication keys.
Exported configuration for [DhcpV4Server tn0-s1]
```

## 2.10.4

### Sample dhcpkey.conf

The following is a sample `dhcpkey.conf` file. Do not directly update this file. Update it only as described in Section 2.10.2 Configuring Authentication Keys on page 26.

```
# Authentication keys file for DHCPv4 Server tn0-s1
# This file was generated by IPWorks at 05/10/05 13:56:12
# It should not be edited directly because any changes
# will be overwritten by IPWorks the next time the
# servers configuration is updated in IPWorks.
#
key {
    key_id 1;
    secret "0x0b0b0b0b0b0b0b0b0b0b0b0b0b0b0b0b";
    client-identifier "ISCDhcpClient";
}
key {
    key_id 2;
    secret "Jefe";
    client-identifier 01:00:50:BA:7F:5C:25;
}
```



## 2.11 User-Defined Options

There are situations where the DHCP option needs to be configured. The IPWorks DHCP server includes a mechanism to configure unsupported options. Using this tag requires the knowledge of knowing the numeric tag associated with the option in which it needs to be configured. All standard options in DHCP have a numeric tag associated with them. This tag is the number that is sent in the DHCP packets when the options are sent to a client.

Once the numeric tag is known for the option to be configured, the value for that option can be set by using the custom option. This is a special option that has two arguments, which are separated by white space: the first argument is the numeric tag, the second argument is the value to send to the client. This value must be specified in hex.

In this example, a custom option is configured and the relevant portion of the configuration file is examined to see how it modifies the servers configuration.

```
IPWorks> select dhcpserver dhcp0
Selected 1 object(s).
IPWorks> modify -add option="custom 124 01:02:03:04"
Working on 1 object(s).
1 object(s) were updated.
IPWorks> list -format=conf

# Configuration file for DHCP Server dhcp0
... <irrelevant lines removed>...
option option-124 01:02:03:04;
```

## 2.12 P-CSCF Discovery

The Proxy-Call Session Control Function (P-CSCF) discovery is triggered by a DHCPINFORM message. In IPWorks, DHCPv4 server uses the relay agent information in the DHCPINFORM message as the criteria to select a P-CSCF server.

The P-CSCF IP address is configured in the DHCP/NACF configuration file (dhcpd.conf).

The example below creates `clientclass` object and specifies the end-user based on the matched criteria:

```
IPWorks> create clientclass P_CSCFclients -set match="if
substring(option agent.circuit-id,0,6) = \"P_CSCF\""
1 object(s) created.
IPWorks> list -format=conf

class "P_CSCFclients" {
match if substring (option agent.circuit-id, 0, 6) = "P_CSCF";
}
```



Since the IP address is allocated by DHCP/NACF, the P-CSCF address can be associated with a specific address pool. The example below defines the P-CSCF option in a specific pool and makes the client class available in this pool:

**Note:** Before using the following command, a subnet such as 10.170.1.0/24 must be created.

```
IPWorks> create pool pool0 -set addressrange=1
0.170.1.1-10.170.1.251 -set server=dhcp0 -set
allowedclient=P_CSCFclients
1 object(s) created.
IPWorks> list -format=conf

# Pool pool0
pool {
range 10.170.1.1 10.170.1.251;
allow members of "P_CSCFclients";
}
```

With the above configuration examples, a given P-CSCF IP address is allocated to a specific range of end-users.

If DHCP encounters data corruption (It cannot analyze or understand the option 120.), it skips the Session Initiation Protocol (SIP) servers handling procedure, and continues with the normal DHCPINFORM procedure. No error message is included in DHCPACK message.

## 2.13 User ID Provisioning

The User ID is the sub-option subscriber ID of Option82 included by the DHCP Relay Agent. In IPWorks, the client can get the User ID from DHCPv4 server if this User ID provisioning function is enabled.

The user can specify a user-defined option for DHCPv4 server to send the User ID to the client. The option range for User ID provisioning is from 224 to 254, and this option is named as `UserID_Prov`.

In the following example, the `UserID_Prov` option is set to 234, which means the option 234 is used to send the User ID to client from DHCPv4 server.

```
IPWorks> modify dhcpv4server dhcp0 -add option=
"custom 234 "UserID_Prov"
Working on 1 object(s).
1 object(s) created.
```



**Note:** DHCPv4 server does not verify the User ID format, it sends the User ID to client as same as it received from DHCP Relay Agent.

## 2.14 Server Reconfiguration

When changes are made to the DHCPv4 Server, the client leases should be renewed with new configuration information. The `reconfig` command is used to make the server notifies clients about this, including changes regarding the address range. DHCPv4 reconfiguration is based on RFC 3203.

The `reconfig` command instructs the server to unicast a `FORCERENEW` message to the clients instructing them to renew their leases: when the clients receive these messages, they request for renewal of leases instantly.

The `reconfig` command can be used from the CLI - to trigger the sending of `FORCERENEW` messages to all the clients that were being served before the server shutdown and whose leases are active after bringing up the server.

In order to have reconfiguration, the server needs to be stopped and restarted after changing the configuration. If, for example, a client's lease is active until 21.00 hours and the server configuration occurs at 20.30 hours and the server is started at 21.10 hours, then a message is not sent. Also, if a client's lease expired at 20.25 hours itself, before shutting down the server at 20.30 hours for changes, even that does not come under the influence of the `reconfig` command.

The `reconfig` command applies to clients, as shown in the above example, that:

- were active at the time of shutdown (that is, at 20.30 hours) and
- the leases are active till later than the restart (that is, > 21.10 hours).

Reconfiguration, thus, proceeds as follows:

1. The `reconfig` command from the CLI is issued.
2. The DHCPv4 Server unicasts `FORCERENEW`. The messages are sent to those clients that were using the server before the shutdown.

When new clients request for a lease, the new configuration information is anyway provided in the lease.

3. Clients request for renewal of lease instantly.
4. The client is updated with the changes in configuration while the lease is renewed.
5. Retransmission of the `FORCERENEW` command is attempted 3 times.

**Note:** Even if the lease is valid until much longer, the client requests renewal instantly. Usually, a renewal request is sent at the half point of the lease duration.



The `reconfig` command can be used only once after changing server configuration and bringing it up. It cannot be issued again before the server is shut down and restarted.

### 2.14.1 Retransmission and Reissuing Reconfig

The `FORCERENEW` messages are retransmitted three times in case the client does not respond. That is why issuing the `reconfig` command more than once within a server lifetime is not allowed.

### 2.14.2 Reconfig Threshold

The `reconfig` threshold specifies the number of `FORCERENEW` messages sent at once. By default (with the value as 0), all `FORCERENEW` messages are sent at once. This causes the network load to peak: when a suitable threshold is specified, batches of messages are sent in a staggered manner.

The number of messages is specified by the `reconfig` threshold and the interval between batches is determined by the system based on the specified threshold. The use of network resources is steady when messages are staggered into batches in this manner.

The threshold is specified in the attribute `reconfigThreshold` of `MO DHCPv4Service`.

```
>ManagedElement=<Node Name>,IpworksFunction=1,IPWorksDHCPProt=1,DHCPv4Service=1
```

```
(DHCPv4Service=1)>configure
```

```
(config-DHCPv4Service=1)>reconfigThreshold=0
```

```
(config-DHCPv4Service=1)>commit
```

```
(DHCPv4Service=1)>
```

### 2.14.3 Show Reconfiguration Status

The `show reconfigstatus` command displays the number of clients that acknowledged the `FORCERENEW` message and the total number of clients that the server is attempting to reach.

This command can be used from the CLI.

### 2.14.4 Auto Reconfigure

When the `enableAutoReconfig` option is selected, the administrator need not issue the `reconfig` command. When the `enableAutoReconfig` option



is enabled in the MO *DHCPv4Service*, DHCPv4 automatically sends the *FORCERENEW* message when there is a change in the address range and the address assigned to the client lease is no longer valid.

Address range change is a critical situation that can result in an IP address conflict, it is good to automatically send the *FORCERENEW* messages to the clients by enabling the *enableAutoReconfig* option.

```
>ManagedElement=<Node Name>,IpworksFunction=1,IPWorksDHCPProtocol=1,DHCPv4Service=1
```

```
(DHCPv4Service=1)>configure
```

```
(config-DHCPv4Service=1)>enableAutoReconfig=true
```

```
(config-DHCPv4Service=1)>commit
```

```
(DHCPv4Service=1)>
```

**Note:** If the self *enableAutoReconfig* is enabled and server configuration is changed in the address range, then the server will unicast the *FORCERENEW* message to the clients that are using invalid leases as per the current configuration of the server. These are the leases server allocated to the clients before undergone through the change in the configuration. In the other hand, on issuing of *reconfig* command, the server will send *FORCERENEW* message no matter what the changes are.



## 3 DHCP Operations

This section describes common operation instructions for DHCP of IPWorks.

### 3.1 Starting and Stopping Servers

IPWorks provides mechanisms for controlling a server once it is installed and operating. Command `ipw-ctr` is used to start and stop the server from the system.

The example that follows shows how to stop and start a server.

### 3.2 Updating Servers

Any configuration change made by the CLI is not activated until updating the server and restarting service. This allows the changes to be undone, or to prepare a set of changes to be made all at once before they are activated. However, it also requires that the server is updated once the changes are made.

If a DHCP server is configured for failover, then any time that server is updated its peer will also be updated. This is because the failover protocol depends on the fact that both servers are using the same configuration, so changes to one require changes to the other. This is true for both the primary and secondary server.

In this example, a single DHCP server that has a failover, peer is automatically updated as well is created.

```
IPWorks> select dhcpserver parilli
Selected 1 object(s).
IPWorks> update
Result of performing an export is:
Exported configuration for [DhcpV4Server parilli]
Result of performing an update on the peer server is:
  Result of performing an export is:
  Exported configuration for [DhcpV4Server blue]
  Updated the configuration and started
    'DHCPV4' server 'blue'.
Updated the configuration and started
  'DHCPV4' server 'parilli'.
```

**Note:** This message is displayed when the Server Manager is started and update is issued. If the update command is issued when the server is running, the following message is displayed:

```
Updated the configuration for 'DHCPV4' server 'parilli'
```



One additional feature of IPWorks is that it is not needed to know all the servers that are affected while making a configuration change. However, it is still important to make sure that these servers get updated when the changes are made, otherwise they will not be correctly activated in the network. To resolve this problem, each server has a field on it called `ExportNeeded` and this is automatically set to true by IPWorks when any aspect of that servers configuration is changed.

The `ExportNeeded` field can be used to very easily update all the servers that have pending changes with the command used in the following example. In this example, only one server needed to be updated. Notice that when the command is repeated, the second time it does not update any servers because the flag has been reset and nothing else has changed.

```
IPWorks> update dhcpservers -where exportneeded=true
Result of performing an export is:
Exported configuration for [DhcpV4Server parilli]
Result of performing an update on the peer server is:
  Result of performing an export is:
  Exported configuration for [DhcpV4Server blue]
  Updated the configuration for 'DHCPV4' server 'blue'.
  Updated the configuration for 'DHCPV4' server 'parilli'.
IPWorks>
```

```
IPWorks> update dhcpservers -where exportneeded=true
No matching object(s) found.
```

### 3.3 Placing Server in Partner-Down Mode

If one of the failover partners needs to be taken offline for an extended time period for system or other maintenance, after shutting down that server, place the other failover server into partner-down mode so it can access the entire lease pool (after the maximum client lead time expires).

IPWCLI (partnerdown command) is used to do the operation. For example:

```
IPWorks> select dhcpserver dhcp2
```

```
Selected 1 object(s).
```

```
IPWorks> partnerdown
```

```
The DHCP server 'dhcp2' was set to the partnerdown state.
```

```
IPWorks>
```

When the offline server is ready to back online, ensure that the return of the server is done properly so that it can synchronize with the running server. Start the offline server while the server previously set to partner-down mode is running and ensure that no network problems prevent communication. In



this way, the two servers can synchronize properly and address assignments are done safely.

If the offline server is back online and cannot communicate with the server that was in partner-down mode, it can resume leasing activity but both servers could assign the same address to different clients.

## 3.4 Searching for Lease

Since the lease objects are not stored in the central database, the user must specify the DHCP server to search.

To accomplish the searches, allow a specification of a source to be as an additional argument for searching in IPWCLI. Once the DHCP server is specified, the search request will then be sent directly to the server.

The following example searches and displays lease objects from an active DHCP server.

```
IPWorks> list lease 10.0.0.4 -source=parilli
```

```
[Lease 10.0.0.4]
Partition: active
Server: parilli
Address: 10.0.0.4
ClientIdentifier: \000zz042f\000\000
HardwareAddress: 01:00:7A:7A:30:34:32:66:00:00
State: free
StartTime: 2006/05/28 14:19:09
EndTime: 2006/05/28 14:20:47
```

To search more than one lease object, use the following command:

```
IPWorks> list lease -source=parilli
```

**Note:** The maximum search range of lease objects is 256. The following message is displayed if the search range exceeds the maximum value:

"Only 256 search results are displayed, please specify the search range in details or change the DHCP Server Manager Parameter '*maxLeaseQuery*' in ECLI."

To specify the search range, use the following command as an example:

```
IPWorks> list lease -source=parilli -where
address>10.0.0.1&&address<10.0.0.100
```





## Reference List

### **Ericsson Documents**

- [1] *IPWorks Configuration Management*
- [2] *IPWorks DHCP Parameter Description*
- [3] *Managed Object Model (MOM)*
- [4] *License Management*
- [5] *Trademark Information*
- [6] *Typographic Conventions*
- [7] *Glossary of Terms and Acronyms*